TAMPEREEN TEKNILLINEN YLIOPISTO

JUSSI RASKU

# TILASTOLLINEN PIIRRELUOKITTELIJA KONENÄKÖLAADUNVALVONNASSA

Diplomityö

Tarkastaja: Robert Piché
Tarkastaja ja aihe hyväksytty
Tieto- ja sähkötekniikan
tiedekuntaneuvoston
kokouksessa 3.2.2010

# TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
**RASKU, JUSSI: Tilastollinen piirreluokittelija konenäkölaadunvalvonnassa**
Diplomityö, 84 sivua, 32 liitesivua
Maaliskuu 2010
Pääaine: Teknillinen matematiikka
Tarkastajat: Professori Robert Piché
Avainsanat: Hahmontunnistus, tilastollinen luokittelija, konenäkölaadunvalvonta

Konenäkölaadunvalvonnalla pyritään takaamaan valmistettavien tuotteiden ja tuotannon laatu. Automatisoitu virheidentunnistus ja luokittelu parantaa laadunvalvonnan luotettavuutta, vaikeiden ongelmien ratkaisukykyä ja prosessin seurantaa. Käyttämällä tilastollista luokittelijaa lisätään konenäkölaadunvalvontajärjestelmän käyttökelpoisuutta laaduntakaamisprosessin työkaluna. Virheen tyypin luotettava tunnistaminen on kuitenkin osoittautunut ongelmalliseksi, minkä vuoksi useat alan tutkimukset keskittyvät yhden tietyn tuotteen tarkastamisessa esiintyviin luokitteluongelmiin. Tässä työssä esitelty luokittelijaohjelmisto tarjoaa menetelmiä konenäkölaadunvalvonnassa esiintyvien luokittelutehtävien ratkaisemiseen.

Työllä oli kolme tavoitetta. Ensimmäisenä tavoitteena oli esitellä konenäkölaadunvalvonnan ammattilaisille joukko käyttökelpoisia visuaalisten virheiden luokittelumenetelmiä. Toisena tavoitteena oli tutkia, miten nämä menetelmät soveltuvat erilaisiin konenäkölaadunvalvonnan luokittelutehtäviin. Kolmas tavoite oli tuottaa työn aikana tilaajan olemassa olevien järjestelmien kanssa yhteensopiva luokittelukirjasto. Toisen ja kolmannen tavoitteen saavuttamiseksi valittiin toteutettavaksi ne luokittimet, joiden ennakoitiin soveltuvan käytettäväksi konenäkölaadunvalvonnassa.

Kirjallisuudesta valittuja testiaineistoja käytettiin toteutettujen luokittimien oikean toiminnan tarkistamiseen. Konenäkölaadunvalvonta-alan testiaineistolla testattiin luokittimien selviytymistä erilaisista laadunvalvontatehtävistä. Luokittimien erot etsittiin vertailemalla virheluokitusten yleisyyttä ja laskennallisia suoritusaikavaatimuksia. Saatujen tulosten avulla analysoitiin toteutettujen tilastollisten luokittimien soveltuvuutta erilaisiin konenäkölaadunvalvonnassa esiintyviin luokitteluongelmiin.

Tämän työn taustalla oli visio kokonaisvaltaisesta laadunvalvontajärjestelmästä, johon luokitin olennaisena osana kuuluu. Luokitin tekee laadunvalvontajärjestelmästä voimakkaan laadun takaamisen työkalun, sillä visuaalisten virheiden luokittelutieto mahdollistaa laadunvalvonnan, tiedonkeruujärjestelmien, raportointijärjestelmien ja ennen kaikkea tuotantoprosessin kehittämisen.

# ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
**RASKU, JUSSI: Statistical pattern classifier in machine vision quality control**
Master of Science Thesis, 84 pages, 32 Appendix pages
March 2010
Major: Mathematics
Examiner: Professor Robert Piché
Keywords: Pattern classification, statistical classifier, machine vision quality control

Machine vision quality control aspires to ensure high quality of the manufactured goods. Automated defect recognition and classification improves reliability and process monitoring capabilities of the quality control system. It also allows tackling more complex quality control problems. By using statistical classification of visual defects the machine vision quality control system can be made even more able tool of the quality assurance. However, reliable recognition of defect type has proven to be difficult task and that probably is the reason why most studies in the machine vision field concentrate only on developing inspection methods for one specific product. The classification software introduced in this thesis gives tools to solve classification tasks in the machine vision quality control field.

This thesis had three objectives. The first one was to present set of practical visual defect classification methods to the reader. Second objective was to study how the selected classification methods were able to solve the classification tasks that rise from the machine vision quality control problems. Third objective was to implement classification software during the course of this thesis that would be compatible with the existing solutions of the customer. To accomplish the second and third objectives a set of classifier methods that showed promise in solving machine vision related problems was chosen.

Test data from the pattern classification literature was used to confirm the right operation of the implemented classifiers. A set of machine vision quality control datasets was used to test the usefulness of the classification algorithms in varied quality control classification tasks. The differences between classification algorithms were sought by comparing misclassification rates and computational requirements. The results made possible to evaluate the suitability of the implemented algorithms in solving classification problems emerging from the machine vision quality control applications.

The vision behind this thesis was seeing the quality control system as a comprehensive process in manufacturing. The classifier is seen as an essential part of the process of producing higher quality goods. Classifier completes the quality control system as a powerful tool of the quality assurance. Visual defect classifier opens up new possibilities in developing data collection systems, reporting systems and above all the manufacturing process.

# ALKUSANAT

Tämän diplomityön tekeminen aloitettiin keväällä 2007 työskennellessäni ZET Systemsillä. Diplomityöni oli tarkoitus olla osa ZET Systemsin konenäkölaadunvalvontaohjelmiston kehitystyötä. Tarkoituksena oli kehittää ohjelmistoa lisäämällä siihen visuaalisten virheiden luokittelumahdollisuus. Työn tekeminen päätoimen ohessa osoittautui kuitenkin haastavaksi. Jätin ZET Systemsin vuodenvaihteessa 2008-2009 muutettuani Jyväskylään, mutta jatkoin työn tekemistä Jyväskylästä käsin. Tänä aikana ZET Systems uudistui jatkaen toimintaansa Grenotechin nimellä. Työn intensiivisin vaihe oli talvella 2009-2010. Työ valmistui lopulta Grenotechin kanssa yhteistyössä alkukeväästä 2010.

Haluan kiittää vaimoani ja perhettäni tuesta, jota olette antaneet myös vaikeina aikoina. Kiitokset ZET Systemsin työyhteisölle, erityisesti Mikko Koivuniemelle, joilta sain kipinän tämän työn tekemiselle.

Suuri kiitos työn tarkastajalle Robert Pichélle, joka otti työn tarkastusvastuun tilanteessa, jossa diplomityön valmistuminen oli epävarmaa. Tapaamisemme lisäsivät luottamusta siihen, että olen oikealla tiellä.

Erityiskiitos Jyväskylän Yliopiston laskennallisen logistiikan tutkimusryhmän jäsenille, jotka olivat korvaamaton apu työn loppuvaiheen aikana.

Jyväskylässä 18.3.2010

# SISÄLLYS

# TERMIT JA LYHENTEET

| | |
|---|---|
| **$\nu$-SVC** | Tyypin 2 Tukivektorikoneluokitin. |
| **ANN** | Lähinaapureiden nopean etsinnän ohjelmistokirjasto. |
| **BBD** | Balanced Box-Decomposition Tree -esiprosessointimenetelmä. |
| **C++** | Ohjelmointikieli. |
| **CSVML** | Tyypin 1 SVM lineaarisella kernelillä. |
| **CSVMR** | Tyypin 1 SVM RFB-kernelillä. |
| **FB** | Sumea luokitin kellokäyrän (bell curve) muotoisilla sumeilla joukoilla. |
| **FCC** | Sumea luokitin (Fuzzy Control Classifier). |
| **FCL** | Tiedostoformaatti (Fuzzy Control Language). |
| **FFLL** | Sumean säätimen toteuttava ohjelmistokirjasto (the Free Fuzzy Logic Library). |
| **FT** | Sumea luokitin kolmion muotoisilla (triangular) sumeilla joukoilla. |
| **FZ** | Sumea luokitin puolisuunnikkaan muotoisilla (trapezoid) sumeilla joukoilla. |
| **k-NN** | Lähinaapuriluokitin (k-Nearest-Neighbour Classifier). |
| **Kohde** | Näytteestä löydetty alue, joka on relevantti laadunvalvonnan kannalta. Tyypillisesti virhe. |
| **Laadunvalvonta** | Tuotannon seurantamenetelmä jonka avulla pyritään takaamaan, että valmistettava tuote vastaa asiakkaan tarpeita ja odotuksia. |
| **LIBSVM** | SVM-luokittimet toteuttava valmiskirjasto. |
| **LP** | Luokkapari. |
| **Luokitin** | Matemaattinen menetelmä, joka määrittää luokittelutuloksen piirrevektorin perusteella. |
| **Luokittelu** | Kohteen määrittäminen tiettyyn luokkaan kuuluvaksi sen piirteiden ja sovelluskohtaisen tietämyksen avulla. |
| **Luokka** | Kategoria, johon kohteen voidaan katsoa kuuluvan. Luokitus on yksikäsitteinen eli kohde voi kuulua vain yhteen luokkaan. |
| **MAP** | Suurin posterioritodennäköisyys (Maximum a Posteriori Probability). |
| **MLP** | Neuroverkkoluokitin (Multi-Layer Perceptron). |
| **MDL** | Lyhimmän kuvauksen periaate (Minimum Description Length). |
| **MOGH** | Usean asteen gradienttihistogrammi (Multiple Order Gradient Histogram). |
| **NB** | Naiivi Bayes -luokitin. |
| **NBD** | Naiivi Bayes -luokitin diskretisoiduilla jakaumaestimaateilla. |
| **NBK** | Naiivi Bayes -luokitin ydinmenetelmän jakaumaestimaateilla. |
| **NBN** | Naiivi Bayes -luokitin normaalijakaumaestimaateilla. |
| **NLCA** | Epälineaarinen komponenttianalyysi. |
| **NP** | Epädeterministisellä Turingin koneella polynomiaalisessa ajassa ratkeavien ongelmien joukko. |

| | |
|---|---|
| **nu-SVM** | katso $\nu$-SVC. |
| **nuSVML** | $\nu$-SVM (tyypin 2 SVM) lineaarisella kernelillä. |
| **nuSVMR** | $\nu$-SVM (tyypin 2 SVM) RFB-kernelillä. |
| **Näyte** | Laadunvalvonnan tarkastettavana oleva tuote tai kappale. |
| **PCA** | Pääkomponenttianalyysi. |
| **Piirre** | Kohteelle laskettu tunnusluku, joka kuvastaa jotain kohteen ominaisuutta. Esimerkiksi kohteen koko on tällainen ominaisuus. |
| **Piirreilmaisin** | Piirteitä kohteelle laskeva konenäköalgoritmi. |
| **Piirrevektori** | Kohteelle lasketuista tunnusluvuista muodostettu vektori. |
| **QA** | Laaduntakaaminen (Quality Assurance). |
| **QC** | Laadunvalvonta (Quality Control). |
| **RBF** | Gaussin käyrän yleistys N-ulotteiseksi (Radial Basis Function). |
| **SMO** | Dekompositiomenetelmä (Sequental Minimal Optimization). |
| **SPC** | Tilastollinen prosessinohjaus (Statistical Process Control). |
| **SVM** | Tukivektorikone (Support Vector Machine). |
| **TSC** | Tarkka ajanottomenetelmä (Time Stamp Counter). |
| **UML** | Ohjelmistojen graafinen mallinnuskieli. |
| **Virhe** | Kohde, jonka tunnusluvut ylittävät kohteille asetetut rajaarvot. |

# SYMBOLIT JA MERKINNÄT

| | |
|---|---|
| $\tilde{A}$ | Sumea joukko. |
| $A_i$ | Luokan $\omega_i$ päätösalue. |
| $C$ | $C$-SVM luokittimen parametri. |
| $c(\mathbf{x})$ | Luokitin. |
| $d$ | Piirreavaruuden dimensio. |
| $D$ | Opetusjoukko. |
| $F_i$ | Piirreilmaisin. |
| $f_{\mathrm{NDF}}(x;\mu,\sigma)$ | Normaalijakauman tiheysfunktio, jonka keskiarvo on $\mu$ ja keskihajonta $\sigma$. |
| $f \sim N(\mu,\sigma^2)$ | $f$ noudattaa normaalijakaumaa parametrein $\mu$ ja $\sigma$. |
| $\gamma$ | RBF-kernelin parametri. |
| $g_i$ | Luokan $\omega_i$ päätösfunktio. |
| $K$ | Kernelifunktio. |
| $l$ | Opetusjoukon alkioiden lukumäärä. |
| $\mu_{\tilde{A}}$ | Sumean joukon $\tilde{A}$ määrittävä jäsenyysfunktio. |
| $\mathbf{M}^{m\times n}$ | $m \times n$ matriisi. |
| $\mathbb{N}$ | Luonnollisten lukujen joukko. |
| $O(f(N))$ | Algoritmin kompleksisuuden asymptoottinen yläraja. |
| $\Omega(\mathbf{x})$ | Kuvausfunktio korkeampidimensionaaliseen piirreavaruuteen. |
| $\omega_i$ | Luokka, jonka tunniste on $i$. |
| $P(A\|B)$ | Tapahtuman A todennäköisyys ehdolla B. |
| $q$ | Luokitustehtävän luokkien lukumäärä. |
| $\mathbb{R}$ | Reaaliavaruus. |
| $\|S\|$ | Joukon S alkioiden lukumäärä. |
| $\nu$ | $\nu$-SVM -luokittimen parametri. |
| $\mathbf{x}$ | Piirrevektori. |
| $X$ | Satunnaismuuttuja tai joukko. |
| $\mathbf{X}$ | Avaruus. |
| $x_i$ | Piirrearvo. |
| $\langle \mathbf{x} \cdot \mathbf{y} \rangle$ | Vektoreiden sisätulo. |
| $y$ | Luokan $\omega_i$ tunniste. |
| $Z_i$ | Virhealueen $i$ tietorakenne. |

# 1. JOHDANTO

Laadunvalvonnalla pyritään takaamaan valmistettavien tuotteiden laatu, vähentämään materiaalihukkaa, tuotantoseisokkeja ja tuotepalautuksia sekä lisäämään asiakastyytyväisyyttä. Laadunvalvonnalla pyritään siis toisaalta varjelemaan tuotteita valmistavan yrityksen mainetta ja toisaalta pienentämään tuotteiden valmistamisen kustannuksia. [47]. Molemmat tavoitteet hyötyvät siitä, että laadunvalvonta tuodaan osaksi tuotantoprosessia. Kun laadunvalvonnan painopistettä siirretään viallisten tuotteiden löytämisestä ja hylkäämisestä laatuongelmien syiden tunnistamiseen ja ymmärtämiseen, voidaan tuotteen laadun lisäksi parantaa koko tuotannon laatua [14, s. 5]. Prosessin ongelmia korjaamalla ja sen toimintaa tehostamalla saadaan kustannussäästöjä sekä valmistetaan parempilaatuisia tuotteita [49, s. 235]. Tuotantoprosessin paremman ymmärtämisen ja ongelmakohtiin puuttumisen edellytyksenä on, että laadunvalvonnalla saadaan kerättyä tarkastettavista kohteista riittävästi oikeanlaista tietoa.

Konenäkölaadunvalvonnan etuna perinteiseen ihmisten suorittamaan laatukontrolliin on sen seurattavuus, toistettavuus, luotettavuus, tarkkuus ja edullisuus [47]. Juuri sen tuomien etujen vuoksi konenäkölaadunvalvonnan kysyntä on taantumasta huolimatta pysynyt korkealla. Konenäköalan liikevaihto Euroopassa vuonna 2008 ylitti EMVA:n [18] mukaan 900 miljoonaa euroa. Tästä liikevaihdosta laadunvalvonnan osuus oli 62.7 % eli noin 565 miljoonaa euroa.

Konenäköpohjaisessa laadunvalvonnassa virheiden kokoja ja sijainteja seuraamalla saadaan kerättyä prosessin tilasta kertovaa tilastotietoa. Kuitenkin tärkeän prosessin ongelmista kertovan suureen, virheen tyypin, tunnistaminen on osoittautunut ongelmalliseksi [4; 63]. Virheen kokoa ja sijaintia voidaan käyttää tarkastettavan kappaleen hylkäysperusteena, mutta tuotantoprosessin ongelmien tunnistamiseen virhetyyppien esiintymistiheydet olisivat hyödyllisempi mittari. Tuotannon ongelmakohtien tunnistaminen puolestaan mahdollistaa tuotantoprosessin kehittämisen. [9]

Tällä työllä on kolme tavoitetta. Ensimmäisenä tavoitteena on esitellä konenäkölaadunvalvonnan ammattilaisille joukko käyttökelpoisia visuaalisten virheiden luokittelumenetelmiä. Toisena tavoitteena on tutkia, miten nämä menetelmät soveltuvat erilaisiin konenäkölaadunvalvonnan luokittelutehtäviin. Kolmas tavoite on tuottaa yleiskäyttöinen luokitteluohjelmisto, jonka työn tilaaja voi sellaisenaan integroida osaksi jo olemassa olevia tuotteitaan.

Työssä vertaillaan olemassa olevia tilastollisten luokittimien teorioita ja analysoidaan niiden soveltuvuutta erilaisiin konenäkölaadunvalvonnassa esiintyviin luokitteluongelmiin. Menetelmiä soveltamalla pyritään automatisoimaan ihmisen tekemää virheiden luokittelu- ja tunnistamistyötä. Menetelmät mahdollistavat myös automatisoidun laadunvalvonnan käyttämisen entistä mutkikkaammissa tarkastustehtävissä. Tällöin tunnistettujen virheiden luokitusta voidaan käyttää apuna tarkastettavan kappaleen hylkäyspäätöstä tehtäessä.

Useat alan tutkimukset keskittyvät vain yhden tietyn tuotteen tarkistamisessa esiintyviin luokitteluongelmiin. Tämän työn tavoitteena on menetelmien yleistettävyys suureen joukkoon konenäkölaadunvalvonnan luokitteluongelmia. Soveltamalla esiteltyjä hahmontunnistuksen menetelmiä virheiden tunnistamiseksi konenäkölaadunvalvontaa tarjoavat yritykset voivat parantaa laadunvalvontaohjelmistojensa yleiskäyttöisyyttä, hyödyllisyyttä ja joustavuutta.

# 2. LÄHTÖKOHDAT

Tässä luvussa kerrotaan mitä tarkoitetaan konenäkölaadunvalvonnalla ja matemaattisella luokittelijalla. Nämä käsitteet muodostavat tämän työn teoreettisen viitekehyksen. Lisäksi esitetään tilastollisen luokittelun teoreettiset lähtökohdat siltä osin kun se on työn kannalta olennaista. Konenäkölaadunvalvontaa käsitellään aliluvuissa 2.1 ja 2.2. Työn tekemisen lähtökohtana on virheiden luokittelun hyödyllisyys, jota perustellaan aliluvussa 2.3. Aliluvussa pyritään osoittamaan luokittelun tärkeä rooli korkeatasoisen tuotannon takaamisessa. Seuraavat kaksi alilukua 2.4 ja 2.5 käytetään matemaattisen luokittimen esittelyyn ja sille tässä työssä asetettujen vaatimusten määrittelyyn. Viimeisessä aliluvussa 2.6 valitaan tässä työssä toteutettavat luokitinmenetelmät.

## 2.1 Laadunvalvonta

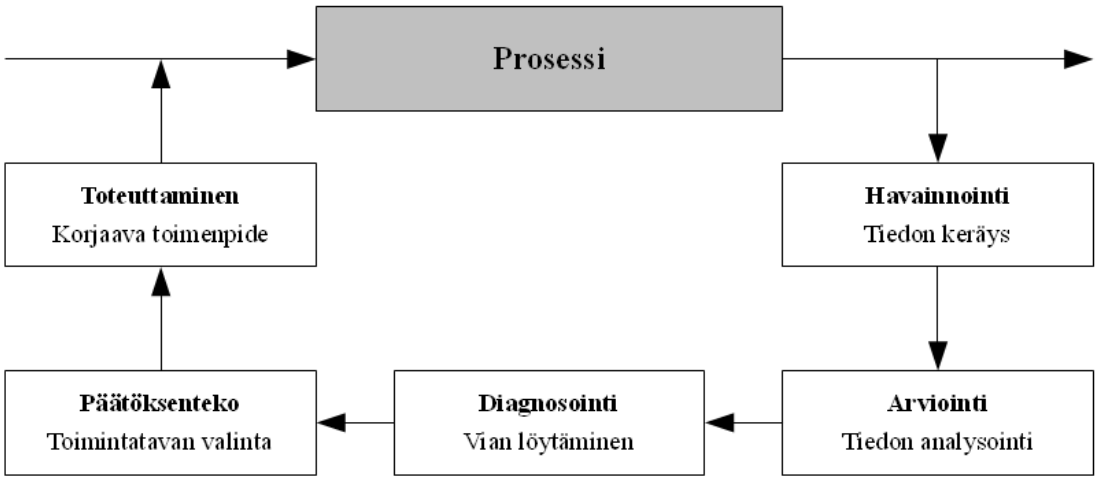
Laavunvalvonnalla pyritään takaamaan, että valmistettava tuote vastaa asiakkaan tarpeita ja odotuksia [66]. Tällä pyritään varmistamaan lopputuotteen riittävä laadukkuus. DeVor et al. [14, s. 11] mukaan perinteinen menetelmä taata lopputuotteiden riittävä laatu on ollut tunnistaa ja poistaa virheelliset tuotteet valmistuksen loppuvaiheessa. Tämä ei ole kuitenkaan tehokasta, ellei virheen aiheuttanutta ongelmaa prosessissa korjata. Näin ollen laadunvalvonnan tulee olla tuotannon ulkopuolisen työkalun sijaan osa tuotantoprosessia. Laadunvalvonnan tavoitteena ei tule olla laadun parantaminen virheellisiä tuotteita löytämällä ja poistamalla, vaan sen tulee olla työkalu, jonka avulla voidaan tunnistaa laadun vaihtelua aiheuttavat tuotantoprosessin ongelmat. [14] Laadunvalvonta (*QC, Quality Control*) tulisikin nähdä ennen kaikkea laadun takaamisen (*QA, Quality Assurance*) työkaluna, jonka avulla tuotantoa voidaan kehittää.

Laadun takaaminen on olennaista silloin, kun laadunvalvonnalla pyritään vähentämään materiaalihukkaa ja saamaan kustannussäästöjä. DeVor et al. [14] ovat muotoilleet laadun ja tuottavuuden välisen yhteyden siten, että laatu ja tuottavuus voivat siirtyä yhdessä oikeaan suuntaan vain silloin, kun keskitytään varsinaiseen tuotantoprosessiin etsimällä ja tunnistamalla syitä prosessin vikoihin ja poistamalla nämä syyt.

DeVor et al. [14, s. 132] ovat tunnistaneet kaksi tapaa käyttää laadunvalvonnasta saatua tilastotietoa prosessin luotettavuuden parantamiseksi:

1. Kerättyä tietoa voidaan käyttää ajoittaisten ja kroonisten tuotantoprosessia haittaavien vikojen tunnistamiseen.

2. Tietoa voidaan myös käyttää indikaattorina prosessin akuutille säätötarpeelle tuotantoprosessin ollessa käynnissä.




***Kuva 2.1.*** *Tilastollisen prosessinohjauksen vaiheet [14].*

Kuvassa 2.1 on kuvattu prosessin ohjauksen takaisinkytkentä, jonka tavoitteena on ylläpitää ja parantaa tuotantoprosessin laatua sekä lisätä sen tehokkuutta. Laadunvalvonnan osuus tästä takaisinkytkennästä on tiedon keräys ja osittain analysointi. Kerätty ja tilastollisin menetelmin analysoitu tieto auttaa diagnosoimaan tuotannon ongelmia, ja kun ongelmien syy on löydetty, voidaan tehdä valistunut päätös toimenpiteistä ongelman korjaamiseksi. [14, s. 134]

Mitä monipuolisempaa aineistoa laadunvalvonnasta saadaan kerättyä, sitä paremmin tuotantoprosessia ymmärretään ja sen olennaisiin ongelmakohtiin voidaan puuttua. Newmanin ja Jainin [49, s. 234] mukaan prosessin ongelmia korjaamalla ja sen toimintaa tehostamalla voidaan saavuttaa laadun paranemisen ja tuotannon tehostumisen kautta huomattavia kustannussäästöjä.

## 2.2 Konenäköjärjestelmän kuvaus

Tyypillisesti konenäkölaadunvalvontajärjestelmä koostuu kappaleenkäsittelyautomatiikasta, valaistuksesta ja sen ohjauksesta, kameroista ja tietokoneella ajettavasta laadunvalvontaohjelmistosta. Laadunvalvontaohjelmisto käyttää kuvien käsittelyyn konenäkökirjastoa, joka tarjoaa kuvien käsittelyyn ja kohteiden etsimiseen tarvittavat konenäköalgoritmit. Konenäkölaadunvalvonnan etu perinteiseen ihmisten suorittamaan laatukontrolliin verrattuna on Mital et al. [47] mukaan sen parempi seurattavuus, toistettavuus, luotettavuus ja tarkkuus.

Tämän työn puitteissa olennaista on konenäköjärjestelmän kyky löytää paikallisia sävyvirheitä bittikartoista. Olemassa olevat konenäkökirjastot tarjoavat menetelmiä tuottaa tunnuslukuja löydetylle kohteelle. Esimerkkinä kuva 2.2, jossa on tasaisen väriseltä taustalta löydetty kaksi ympäristöään tummempaa tahraa, joille on laskettu tunnusluvuiksi niiden pinta-alat (0.069 $mm^2$ ja 0.189 $mm^2$).


***Kuva 2.2.*** *Konenäköohjelmiston löytämiä visuaalisia virheitä.*

Tyypillisesti konenäköohjelmistoilla voidaan tunnistaa ja lajitella virheitä niiden koon ja sijainnin mukaan. Tämän tyyppistä lajittelua tulisi kehittää, sillä olennainen osa tuotantoprosessin tilan seurantaa on erilaisten virhetyyppien esiintymistiheyden tilastointi. Virheiden lajittelu tehdään laatupäällikön laatimien kriteerien mukaisesti. Esimerkiksi sijainnin mukaisessa lajittelussa tarkastettava pinta jaetaan käsin alueisiin, joissa ilmenevä virhe voidaan merkitä kuuluvaksi alueen mukaiseen virhetyyppiin.

Tässä työssä oletetaan, että nykyinen konenäköjärjestelmä tallentaa tunnistetut virheet. Tietokannasta voidaan myöhemmin laatia raportointityökalun avulla raportteja tuotannon laadusta ja virhejakaumista. Järjestelmässä oletetaan myös olevan valmius tallentaa tieto virhetyypistä tietokantaan.

Ensimmäisessä aliluvussa 2.2.1 esitellään konenäkölaadunvalvontaprosessin vaiheet. Toisessa aliluvussa 2.2.2 esitellään virheentunnistuksen toimintaperiaate ja sen osat yleisellä tasolla. Aliluvussa 2.2.3 esitellään piirreilmaisimien käsite ja selitetään niiden merkitys luokittelijalle.

### 2.2.1 Vaiheet

Konenäkölaadunvalvonta voidaan Malamas et al. [44] mukaillen jakaa neljään vaiheeseen: kuvantamiseen (*image acquisition*), kuvankäsittelyyn ja segmentointiin (*image processing and segmentation*), piirteiden irrotukseen (*feature extraction*) sekä piirteiden tulkintaan ja päätöksentekoon (*pattern interpretation and decision making*). Kutakin vaihetta tarkastellaan seuraavaksi yksityiskohtaisemmin.

**Kuvantaminen**. Hyvin suunniteltu ja toteutettu kuvausjärjestely on edellytys onnistuneelle konenäkölaadunvalvonnalle. Kohteen tulee näkyä kuvassa käyttötarpeeseen nähden riittävän tarkasti ja häiriöttömästi. Hyvälaatuinen kuva saavutetaan valitsemalla ja suunnittelemalla kamerat, optiikka, valot, pidikkeet ja mahdollinen mekaaninen liike siten, että tarkastustehtävän kannalta kiinnostavat piirteet näkyvät kuvassa selvästi. Esimerkiksi naarmut ja painaumat saadaan näkyviin sopivalla kollimoidulla eli optisen akselin suuntaisella valolla. Vastaavasti pinnan painokuvio ja sävyvaihtelut saadaan hyvin näkyviin käyttämällä diffusoitua eli pehmennettyä valoa. Kameran kennon tarkkuus, optiikka ja kuva-ala valitaan sovelluskohtaisesti siten, että otetun kuvan resoluutio riittää vaatimusten mukaiseen erottelukykyyn. [63, s. 145]

**Kuvankäsittely ja segmentointi**. Segmentointi tarkoittaa menetelmää, jolla kuvasta irrotetaan jatkokäsittelyä varten tarkistustehtävän kannalta olennaiset alueet. Ennen segmentointia kuvantamisvaiheessa otettua kuvaa voidaan käsitellä ohjelmallisesti. Suodattamalla kuvaa voidaan päästä eroon sellaisista kuvan piirteistä, jotka eivät ole kiinnostavia segmentoinnin kannalta. Vaihtoehtoisesti suodatuksella pyritään saamaan näkyviin laadunvalvonnan kannalta kiinnostavia kohteita. Yhdistämällä eri valoilla otettuja ja eri suodattimilla käsiteltyjä kuvia saadaan tietyt kuvan kohteet paremmin näkyviin.

**Piirteiden irrotus**. Segmentoidun kuvan alueista irrotetaan kohteet esimerkiksi kynnystämällä tapauskohtaisella harmaasävyn raja-arvolla. Tuloksena on yksittäistä kohdetta kuvaava tietorakenne. Tietorakenteiden avulla kohteelle voidaan laskea kvantitatiivista tietoa, kuten kahden kohteen välinen etäisyys, kohteen koko ja reunaviivan pituus. Näitä mittaustuloksia kutsutaan yleisesti piirteiksi (*feature*).

**Piirteiden tulkinta ja päätöksenteko**. Kuvan suuresta bittimäärästä on edellisten vaiheiden avulla saatu jalostettua muutamia tunnuslukuja, joilla kuvataan kuvasta löytyneiden kohteiden ominaisuuksia. Nämä kohteet voivat olla esimerkiksi kappaleesta mitattavia mittoja tai kappaleen pinnan epäsäännöllisyyksiä. Tunnuslukujen avulla voidaan suorittaa laadunvalvontaa annettujen hylkäysrajojen mukaisesti. Tulkitsemalla tunnuslukuja ja seuraamalla niiden muuttumista ajan kuluessa saadaan tietoa tuotantoprosessin tilasta. Jotta tuotantoprosessin tila ja mahdolliset ongelmat saadaan tunnistettua, on otettava käyttöön tilastollisia menetelmiä. [14] Yksi tällainen menetelmä on virhetyypin päättely tarkastettavasta kappaleesta luokittimelle syötettyjen tunnuslukujen perusteella.

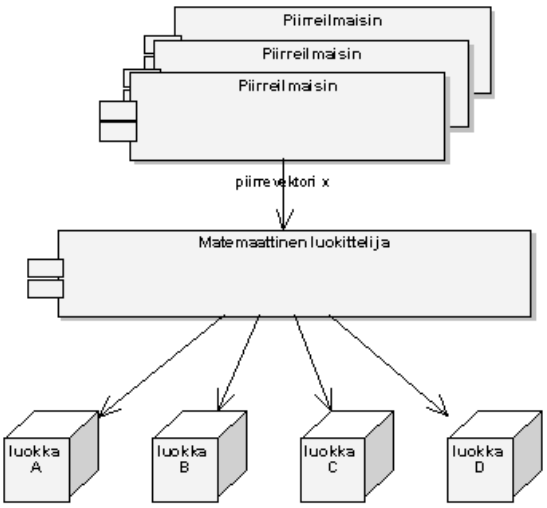
### 2.2.2 Virheentunnistus

Virheentunnistuksen tavoitteena on tunnistaa havainnoissa mahdollisesti esiintyviä virheitä. Tunnistaminen edellyttää, että kukin löydetty virhe pystytään luokittelemaan kuuluvaksi johonkin tunnettuun virhetyyppiin. Virheentunnistus toimii pro-

sessina, joka voidaan jakaa vaiheisiin mukaillen Duda et al. [16, s. 9-13] esittelemää hahmontunnistusjärjestelmien yleistä rakennetta:

1. Havaintojen keräys
2. Tiedon pelkistys
3. Tiedon kuvaus
4. Tiedon luokittelu
5. Päätöksenteko

Virheentunnistusprosessi aloitetaan keräämällä havainnot ja muokkaamalla ne muotoon, josta virheentunnistuksen kannalta kiinnostavat kohteet on mahdollista löytää. Havainnot ovat digitaalisin kuvantamisvälinein saatuja bittikarttoja, joita on tarvittaessa käsitelty kuvankäsittelymenetelmin ja suodattimin. Bittikartat ovat tyypillisesti harmaasävykuvia, mutta myös värikuvia voidaan käyttää, jos se on sovelluksen kannalta välttämätöntä.




***Kuva 2.3.*** *Virheentunnistuksen osat*

Havaintoja pelkistetään segmentoimalla kuva pienempiin alueisiin. Tyypillisesti yksi alue vastaa yhtä laadunvalvonnan kannalta kiinnostavaa kuvasta löydettyä kohdetta. Piirreilmaisimet tuottavat kohteen tietorakenteesta piirrevektorin. Piirrevektori sisältää kyseistä kohdetta kuvaavia tunnuslukuja, kuten esimerkiksi pituus ja pinta-ala. Luokitin saa syötteekseen tämän piirrevektorin ja määrittää luokan, johon kyseinen kohde tunnuslukujensa perusteella kuuluu. Tämän jälkeen konenäköjärjestelmä voi käyttää löydetyn kohteen luokkatietoa päätöksenteossa. Tiedon pelkistyksessä käytettävät piirreilmaisimet ovat luokittimesta erillisiä konenäköalgoritmeja,

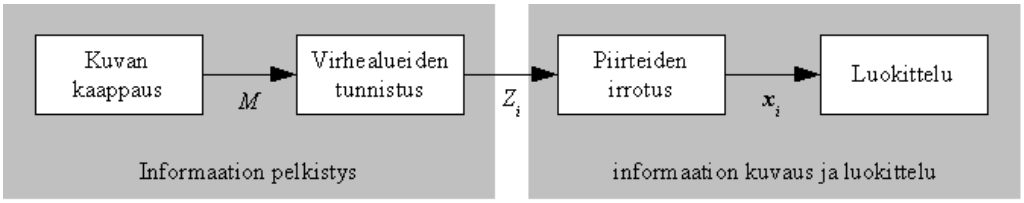
joiden tehtävänä on pelkistää havainnot luokittimen tarvitsemaan muotoon. Tätä jakoa havainnollistetaan kuvassa 2.3.

Luokat on tyypillisesti määritetty käsin tai opetettu luokittimelle opetusaineiston avulla. Luokkien opetustapa riippuu virheentunnistukseen valitusta luokittelumenetelmästä. Luokkien ja piirreilmaisimien lukumäärää ei tässä sovelluksessa ole tarpeen rajoittaa. Erilaisia virhetyyppejä on monimutkaisimmassakin laadunvalvontatehtävässä korkeintaan muutama kymmenen ja piirreilmaisimia tyypillisesti vähemmän kuin viisi.

Työssä keskitytään visuaalisten virheiden luokitteluun. Visuaalisella virheellä tässä työssä tarkoitetaan kappaleen pinnassa olevia virheitä, jotka voivat olla joko tahroja tai sitten erilaisia pinnan epäsäännöllisyyksiä (esimerkiksi koloja ja naarmuja).

### 2.2.3 Piirreilmaisimet

Konenäköpohjaisessa laadunvalvonnassa kameralta saatavat bittikartat ovat usein pikselimäärältään suuria. Esimerkiksi JAI A1 -kamera tuottaa $1,4$ megapikselin bittikarttoja, jolloin yksi kuva on $1392 \times 1040$ matriisi, jonka alkiot ovat kokonaislukuja väliltä [0-255]. Kukin alkio kertoo kyseisen pikselin harmaasävyarvon. Tällaisenaan informaation määrä on liian suuri käytettäväksi tilastollisen luokittimen syötteenä. Informaatiota on ensin jalostettava etsimällä tehtävän kannalta kiinnostavat kohteet ja kuvaamalla ne tunnusluvuiksi luokitinta varten.




***Kuva 2.4.*** *Kuvainformaation jalostus luokittelutuloksiksi*

Informaation jalostus tapahtuu kahdessa vaiheessa. Nämä vaiheet ovat kuvainformaation pelkistys ja informaation kuvaus piirrevektoriksi (Kuva 2.4).

Ensin bittikarttamatriisista $\mathbf{M}^{m \times n}$ etsitään virhealueet käyttäen konenäkölaadunvalvontaohjelmiston kykyä löytää paikallisia sävyvaihteluita bittikartoista. Menetelmien sisältöön ei oteta kantaa, sillä tämän työn osalta riittää, että tällaisten menetelmien oletetaan olevan käytettävissä.

Löydetyt kohteet kuvataan ohjelmiston sisäiseksi tietorakenteeksi $\mathbf{Z}_i$. Käytetty virheiden etsimismenetelmä riippuu sovelluksesta. Esimerkkinä kuva 2.2 sivulla 5, jossa on tasaisen väriseltä taustalta löydetty kaksi ympäristöään tummempaa tahraa (kooltaan 0.069 mm$^2$ ja 0.189 mm$^2$).

Toisessa vaiheessa tunnistetuille virhealueille suoritetaan virhetiedon kuvaus tunnusluvuiksi eli piirteiden irrotus (*feature extraction*). Piirteiden irrottamisen suorittavat piirreilmaisimet. Piirreilmaisimet saavat syötteekseen yksittäistä virhealuetta kuvaavan tietorakenteen $\mathbf{Z}_i$. Tämän tietorakenteen avulla piirreilmaisimet tuottavat virheen tunnuslukuja.

Piirreilmaisimien generoimaa vektoria $\mathbf{x}$ kutsutaan piirrevektoriksi. Kukin valittu piirreilmaisin tuottaa kohdetta kuvaavasta tietorakenteesta $\mathbf{Z}_i$ yhden alkion $x_j$ piirrevektoriin $\mathbf{x}$. $x_j$ voi kuvata esimerkiksi, kuinka suuri tai verkottunut löydetty virhealue oli.

Formalisoituna valitut piirreilmaisimet $F_i \in \mathcal{F}, i = 1, \ldots, d$ kuvaavat parametrina annettavan virhealueen $\mathbf{Z}_i$ reaalivektoriksi $\mathbf{x} \in \mathbb{R}^d$, kun merkitään käytettävää piirreilmaisimien lukumäärää luvulla $d \in \mathbb{N}^+$. Tässä $\mathcal{F}$ on joukko, jonka muodostaa kaikki valittavissa olevat piirreilmaisimet.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} = \begin{bmatrix} F_1(Z) \\ F_2(Z) \\ \vdots \\ F_d(Z) \end{bmatrix} \tag{2.1}$$

Piirreilmaisimet $F_j$ siis jalostavat kohteen tietorakenteen $\mathbf{Z}_i$ reaali(piirre)vektoriksi $\mathbf{x}_i$. Piirrevektori sisältää nyt kohteesta lasketut tunnusluvut, jotka kertovat virheen ominaisuudet.

Käytettävät piirreilmaisimet valitaan aina sovelluskohtaisesti siten, että tunnusluvut kuvaavat mahdollisimman hyvin kohteita. Lisäksi piirreilmaisimien tuottamien tunnuslukujen tulee olla luokan sisällä samankaltaisia ja luokkien välillä selvästi toisistaan eroavia. Oikein valitut piirreilmaisimet takaavat, että havaitut virheet pystytään luokittelemaan luotettavasti.

**Esimerkki.** Sovelluksessa tarkistettava tuote tulee hylätä aina, jos linssialueelta löytyy naarmuja. Sen sijaan pienet tahrat sallitaan (pinta-ala alle 0,2 mm$^2$). Luokittelijaa voidaan käyttää erottamaan löydetyistä alle 0,2 mm$^2$ virheistä tahrat ja naarmut valitsemalla piirreilmaisimiksi virhealueen pituuden, ”pitkänomaisuuden” (leveyden ja pituuden suhteen) ja virhealueiden vierekkäisyyden (mitä lähempänä lähin naapurivirhealue on, sitä suurempi tunnusluku).

Piirreilmaisimien oikea valinta saattaa olla jopa työläin vaihe hahmontunnistusjärjestelmän suunnittelussa. Siksi tämänkin luokittimia vertailevan tutkimuksen kannalta on olennaista tunnistaa hyvä piirreilmaisin. Kunttu on koonnut väitöskirjaansa *Shape and Gray Level Descriptors for Surface Defect Image Retreval and Classification* [38] listan hyvien piirreilmaisimien ominaisuuksia. Lähteenä hän on käyttänyt Carlinin [5] sekä Lin ja Edwardsin [42] julkaisemia artikkeleita. Kuntun

listan ominaisuudet ovat kohteiden muotoa kuvaaville piirteille, mutta ne voidaan yleistää myös kokoa ja sijaintia mittaaviin ominaisuuksiin.

**Riippumattomuus asennosta.** Kahdella samanlaisella kohteella tulee olla sama piirrearvo riippumatta niiden asennosta (ja koosta, mikäli koko ei ole piirrearvo tai osa sitä).

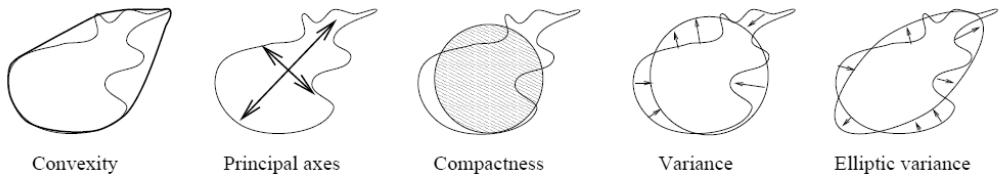
**Yksikäsitteisyys.** Jos kahdella kohteella on eri piirrearvo, ne ovat erilaisia.

**Stabiilisuus.** Jos kaksi kohdetta eroaa vain vähän toisistaan, tulee niiden piirrearvojenkin ero olla pieni.

**Ymmärrettävyys.** Piirreilmaisimen pitää olla jokin laadunvalvonnassa tyypillisesti mitattava suure.

**Kääntyvyys.** Piirrearvon tulee kertoa kohteesta jotain, ja kääntäen kohteelle tulee pystyä laskemaan sen piirrearvo.

**Tehokkuus.** Erityisesti reaaliaikaisissa on-line -konenäköjärjestelmissä on tärkeää, että piirrearvo on nopea laskea kohteen tietorakenteesta.




***Kuva 2.5.*** *Esimerkkejä visuaalisille virheille laskettavista piirteistä [29].*

Hyviä lähteitä visuaalisten virheiden luokitteluun soveltuvien piirreilmaisimien valintaan ovat Iivarinen ja Visa [29] sekä Kunttu [38]. Lähteissä esitellään useita epäsäännöllisen muotoisten kohteiden piirteiden laskentamenetelmiä. Kokoomateoksessa Pietikäinen et al. [54] on useita artikkeleita, joissa käsitellään piirteiden laskemista kuvioiduille pinnoille. Kuvassa 2.5 on muutamia tällaisia esimerkkejä piirremitoista.

## 2.3 Luokittelulla saavutettavat hyödyt

Tarkastettavan tuotteen hyväksymis- ja hylkäyspäätöksiä tekevät konenäkölaadunvalvontajärjestelmät keräävät päätöstä varten monenlaista tietoa tarkastettavasta kohteesta ja siitä mahdollisesti löydettävistä virheistä. Löydetylle visuaaliselle virheelle lasketaan tyypillisesti geometrisia mittoja, kuten pinta-ala, keskimääräinen halkaisija ja symmetrialävistäjä. Tunnuslukujen avulla tehdään päätös siitä, täyttääkö tarkastettava kappale tuotteelle asetetut laatuvaatimukset [63]. Tunnuslukuja tyypillisesti myös tilastoidaan tuotantoprosessin tilan seuraamiseksi [14].

Asetettujen raja-arvojen perusteella tehtävä laadunvalvonta riittää silloin, kun järjestelmää halutaan käyttää kappaleiden hyväksymis-/hylkäämisautomaattina. Tällöin ei kuitenkaan käytetä täyttä konenäkölaadunvalvonnan tarjoamaa potentiaalia. Tuotantoprosessin kannalta virheen tunnuslukuja käyttökelpoisempi tieto on tarkastettavan kappaleen hylkäyksen aiheuttaneiden virheiden tyyppi. Mahdollisuus automatisoidusti tunnistaa virheitä ja luokitella niitä eri virhetyyppeihin parantaa konenäköpohjaisen laadunvalvonnan luotettavuutta, vaikeiden ongelmien ratkaisukykyä ja laadunseurantamahdollisuuksia. Perusteluna työlle on, että virheluokittelija parantaa konenäkölaadunvalvontajärjestelmän käyttökelpoisuutta laaduntakaamisprosessin työkaluna. [9]

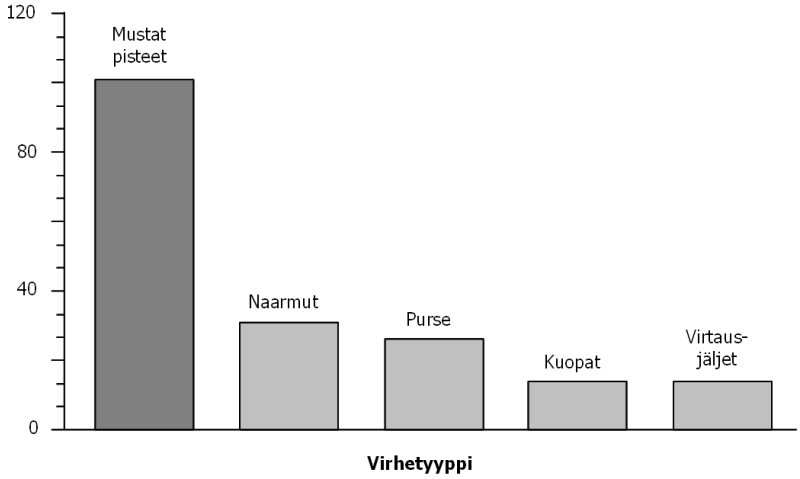
Luokittelijan tehtävänä järjestelmässä on luokitella näytekappaleesta löydetyt virheet esimääriteltyihin luokkiin. Olemassa olevasta konenäköjärjestelmästä voidaan käyttää visuaalisten virheiden etsintäalgoritmit sellaisenaan. Luokittelija ottaa syötteekseen löydetyille visuaalisille virheille tai muille kiinnostaville kohteille laskettuja tunnuslukuja ja päättelee kohteen luokituksen niiden avulla. Luokittelu siis tehdään virheen koon, muodon, sijainnin ja mahdollisesti myös muiden ominaisuuksien perusteella. Visuaalisten virheiden luokittelu on ominaisuus, joka on mahdollista lisätä olemassa oleviin laadunvalvontasovelluksiin jälkikäteen, niin että siitä tulee kiinteä osa laadunvalvontaohjelmistoa. Liitännäinen sisältää varsinaiset luokitinalgoritmit, tuloksen päättely- ja asetusarvomoduulin sekä konfigurointisivun, joka toimii luokittelijan käyttöliittymänä (Kuva 2.7, s. 17).

Luokittelijan tarjoaa muutamia keskeisiä etuja yksinkertaisempaan järjestelmään nähden. Ensinnäkin sen avulla voidaan ottaa monimutkaisempia laadunvalvontatehtäviä kuin aikaisemmin. Luokittelija tarjoaa mahdollisuuden tunnistaa kaikista löydetyistä virheistä ne, jotka johtavat kappaleen hylkäämiseen. Toiseksi luokittelija tuo lisäarvoa tarkemman ja yksityiskohtaisemman tiedonkeruun muodossa. Asiakkaat voivat seurata ja analysoida tuotteen valmistusprosessia entistä tarkemmin, sillä hylkäyksen aiheuttaneiden virheiden sijainnin ja koon lisäksi tarjolla on tietoa myös virheen tyypistä. Tämä tieto mahdollistaa tuotantoprosessin entistä paremman tuntemuksen sekä sen säätämisen entistä tehokkaammaksi. [9]

Luokittelemalla löydettyjä virheitä virhetyyppeihin saadaan jalostettua tietoa, jota voidaan hyödyntää tuotantoprosessin kehittämisessä. Tuntemalla hylättyjen kappaleiden tyypillisimmät virheet ja niiden tyypit voidaan keskittyä olennaisiin tuotannon ongelmiin [9]. Virheiden luokittelu ja luokitusten tilastointi ovat erinomainen tilastollisen prosessinohjauksen (*SPC, Statistical Process Control*) apuväline. Koska luokittelu on pikemminkin ongelmadiagnostiikan kuin tiedon analysoinnin apuväline, luokittelulla täydennetty konenäkölaadunvalvonta pystyy tarjoamaan työkaluja kolmeen kuvan 2.1 prosessinkehitysmallin vaiheeseen; havainnointiin, analysointiin ja diagnosointiin. Perinteinen konenäkölaadunvalvontajärjestelmä kerää ja analysoi

tietoa. Luokittelun lisääminen tekee konenäköjärjestelmästä voimakkaan tuotantoprosessin diagnostiikkatyökalun. Tilastollisen prosessinohjusmallin mukaisesti ihmisen tehtäväksi jää päättää prosessin ongelmia parantava toimenpide ja pistää se käytäntöön. [14, s.132-135]

Tilastoitujen virheluokitusten avulla voidaan laatia Pareto-diagrammi, joka on tilastollisen prosessinohjauksen apuväline [14, s.174]. Pareto-diagrammin hyödyllisyys perustuu Pareton periaatteeseen, jonka mukaan 20 prosenttia syistä johtaa 80 prosenttiin seuraamuksista. Periaatetta ei tule pitää täsmällisenä matemaattisena totuutena, vaan vain käyttökelpoisena oletuksena ja muistisääntönä tosimaailman ilmiöiden luonteesta.




***Kuva 2.6.*** *Pareto-diagrammi erään tapaustutkimuksen ruiskuvalusolun virhetyyppien jakautumisesta. [14, s. 485]*

Hyvä esimerkki virheiden luokittelun hyödyllisyydestä ja Pareto-diagrammin käytöstä on DeVor et al. [14, s. 482-490] raportoima tapaustutkimus muovin ruiskuvaluprosessin parantamisesta SPC-menetelmällä. Kuvan 2.6 tapaustutkimuksessa seurattiin yhtä Ford Motor Companyn Plastic Products Divisionin ruiskuvalusolua. Pareto-diagrammissa on tilastoituna seurantajakson aikana laadunvalvonnassa havaitut virheet tyypeittäin. Diagrammista nähdään, että yli puolet kaikista tilastoiduista virheistä ovat mustia pisteitä. Tapaustutkimuksessa oli saavutettu erinomaisia tuloksia puuttumalla Pareto-diagrammin avulla löydettyihin virhelähteisiin. Prosessin tuotteiden virhetiheys saatiin pudotettua $4,5$ virheestä kappaletta kohden $0,17$ virheeseen.

Virheluokittelu soveltuu hyvin automatisoitavaksi, sillä laaduntarkastuksessa on yleensä olemassa selkeät, luokkarajat määrittelevät säännöt [39]. Silloin kun formaaleja sääntöjä ei ole, vaan luokittelu tapahtuu subjektiivisten päätösten perusteella, voidaan valita luokittelumenetelmät siten, että niillä mallinnetaan luokitusprosessin sijaan luokittelua tekevän asiantuntijan päätöksentekoa.

## 2.4 Luokitin

Luokitin on algoritmi, joka kohteesta kerätyn ja jalostetun tiedon perusteella määrittää kappaleen kuuluvaksi johonkin luokkaan. Luokittimeksi on menestyksekkäästi sovellettu lukuisia matemaattisia menetelmiä. Menetelmiä on kirjallisuudessa jaettu eri tyyppeihin monella eri tavalla. Alla tutustutaan näihin luokitinjaotteluihin. Samalla kartoitetaan, minkä tyyppiset luokittelu- ja opetusmenetelmät sopivat parhaiten laadunvalvontaan.

### 2.4.1 Oppimismenetelmät

Luokitin täytyy opettaa aina luokittelutehtävän mukaan. Riippuen luokittelumenetelmästä ja -tarpeesta oppimistapa kuitenkin vaihtelee. Luokittimet voidaan jaotella eri ryhmiin niiden opettamistavan mukaan. Ryhmittely mukailee lähteitä Duda et al. 2001 [16] ja Cristianini & Shawe-Taylor 2000 [13].

**Ohjaamaton oppiminen**. Kun kohteiden luokkia tai niiden lukumäärää ei tiedetä etukäteen, on käytettävä menetelmää, jossa luokittimen opetusalgoritmi muodostaa itsenäisesti näytteistä toisistaan eroavia ryhmiä. Tämä tunnetaan klusterointiprobleemana.

Tyypillisesti laadunvalvonnassa tiedetään prosessissa esiintyvät virhetyypit jo ennalta ja niiden esiintymistiheydelle voidaan antaa estimaatteja. Automatisoidun virheentunnistuksen kyky valita itsenäisesti luokat ei yleensä ole laadunvalvonnassa tarpeen. Toisaalta automaattisella klusteroinnilla on mahdollista löytää opetusvaiheessa sellaisiakin luokkia, joita ei etukäteen osattu tunnistaa omaksi virhetyypikseen, ja jotka siitä huolimatta tarjoavat arvokasta lisätietoa prosessin toiminnasta.

**Ohjattu oppiminen**. Tässä oppimismenetelmässä luokitin opetetaan opetusaineiston avulla luokittelemaan kohteet esimääriteltyihin luokkiin. Aineisto on joukko kohteita, joiden luokitukset tiedetään. $l$ näytettä sisältävä opetusjoukko voidaan määritellä seuraavasti:

$$D = \left( (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l) \right) \subseteq (\mathbf{X} \times \mathbf{Y})^l \tag{2.2}$$

Vaatimuksena on, että opetusjoukon näytteet tulee luokitella ennen opettamista käsin, joten luokittimen käyttäjän täytyy tuntea luokkien määrä ja ominaisuudet jo ennalta. Tämä oppimis- ja opettamismenetelmä on kuitenkin laadunvalvonnassa kenties käyttökelpoisin, sillä tavallisimmat virhetyypit ja niiden kuvaukset tiedetään tavallisesti luokitinta opetettaessa.

**Vahvistusoppiminen**. Luokitinta ohjataan luokitteluprosessin aikana antamalla palautetta luokittelun onnistumisesta. Palaute korjaa luokitinta toimimaan ohjauksen mukaisesti. Tämäkään menetelmä ei sovi laadunvalvontaan, sillä se vaatii ajonaikaisen laadunvalvonnan tekemisen kahdesti: ensin automatisoidusti tilastollisen luo-

kittimen avulla ja sen jälkeen käsin laadunvalvojan toimesta. Tällöin automaattista luokitinta säädetään siis koko prosessin ajan. Automatisoidun laadunvalvonnan idea vesittyy, mikäli se tarvitsee oman valvontansa.

**Oppimiskyvyttömät järjestelmät**. On tapauksia, joissa hylkäys- ja luokitteluperusteet eivät ole kvantifioitavissa. Tällöin laadunvalvonta perustuu laatutarkkailua tekevän henkilön subjektiiviseen arvioon. Tällaisen laadunvalvontaprosessin mallintaminen automatisointia varten on hyvin vaikeaa. Virheiden luokittelun osalta tällaisten ongelmien luonne estää oppivien menetelmien käytön. Näin voi myös olla silloin, kun opetusaineistoa ei ole saatavilla riittävästi luokittimen opettamista varten. Näissä tapauksissa asiantuntijan tietämystä luokitteluongelmasta ja luokittelusäännöistä voidaan käyttää luokittimen rakentamiseen. Haasteena on kehittää sellainen luokittelumenetelmä, jonka säätäminen on ihmiselle intuitiivista.

### 2.4.2 Laskennalliset tekniikat

Luokittelualgoritmit voidaan jakaa eri ryhmiin myös niissä käytettävien laskennallisten tekniikoiden mukaan. Esitelty jaottelu mukailee Vuoren [69] esittelemää menetelmäjakoa.

**Geometriset menetelmät**. Geometrisissa menetelmissä yhteisenä tekijänä on piirreavaruuden jakaminen osiin hypertasoja käyttäen. Muodostuvat alijoukot muodostavat luokittimen tunnistamat luokat. Esimerkkinä tämän tyyppisistä menetelmistä on aliluvussa 3.5 esiteltävä SVM-luokitin (*Support Vector Machine*). Geometristen luokittimien toimintaperiaate on kohtalaisen helppo ymmärtää ja ne ovat nopeita tekemään luokittelupäätöksiä, vaikka luokitteluongelma olisikin vaikea [13].

**Neuroverkkomenetelmät**. Neuroverkot sopivat joustavuutensa ja opetettavuutensa puolesta hyvin luokittimeksi. Hyvinkin yksinkertaisilla verkoilla voidaan suoriutua lineaarisista luokittelutehtävistä [13, s. 11]. Yhdistämällä neuroneista neuroverkkoja suoriudutaan myös haastavammista epälineaarisista luokittelutehtävistä. Tyypillinen neuroverkkoluokitin on MLP-verkko (*Multi-layer Perceptron*), jossa sisääntulokerroksessa on jokaiselle piirrevektorin tunnusluvulle oma perceptroninsa ja ulostulokerroksessa niin monta perceptronia kuin ongelmassa on kohdeluokkia [48]. Neuroverkkomenetelmät soveltuvat kuitenkin massiivisten opetusaineistovaatimuksiensa, vaikean säädettävyytensä ja vaikeasti esitettävän ”black-box” -rakenteensa takia huonosti konenäköpohjaisen laadunvalvonnan luokittimeksi. Monimutkaiset neuroverkot vaativat myös paljon laskentakapasiteettia. Tätä ei jatkuvissa laadunvalvontaprosesseissa ole tarjolla, sillä päätökset tulee tehdä nopeasti.

**Tilastolliset menetelmät**. Tilastollisessa luokittelussa kohteiden tunnuslukuja tarkastellaan tilastollisina muuttujina. Tyypillisesti pyritään rakentamaan estimaattori $P(\omega_i|\mathbf{x})$, missä $\mathbf{x}$ on piirrevektori ja $\omega_i$ yksi luokittelun kohdeluokista. $P$ estimoi, millä todennäköisyydellä kohde, jonka piirrevektori on $\mathbf{x}$, kuuluu luokkaan $\omega_i$.

Esimerkkinä tällaisesta luokittimesta on aliluvun 3.2 naiivi Bayes -luokitin. Tilastollisten menetelmien teorian avulla voidaan rakentaa luokitin, jonka voidaan osoittaa olevan optimaalinen luokitteluongelman ratkaisu. Tämä kuitenkin edellyttää, että ongelman tilastollisten muuttujien jakaumat tunnetaan. [16, s. 23]. Koska jakaumia ei tosielämän sovelluksissa tunneta, joudutaan tilastollista luokitinta rakentaessa tekemään oletuksia, jotka huonontavat luokittelutarkkuutta. Tilastolliset menetelmät tuntuvat sopivan hyvin laadunvalvontaan, sillä virhetyypit tiedetään yleensä ennalta ja virheiden esiintymistiheyksille voidaan joko laskea estimaatit opetusdatasta tai ne voidaan arvioida käsin. Tilastollisten menetelmien etuna on lisäksi se, että ne ovat yksinkertaisia ja nopeita [3].

**Syntaktiset menetelmät ja rakenteiset menetelmät**. Joissain ongelmissa raakainformaation pelkistäminen tunnusluvuiksi eli piirrevektoriksi $\mathbf{x}$ voi olla vaikeaa tai jopa mahdotonta. Näille menetelmille on yhteistä se, että mikäli luokiteltava kohde voidaan tästä huolimatta esittää merkkijonona tai vaikkapa suunnattuna graafina, tai mikäli ne ovat jo alkujaan jossain tällaisessa muodossa, voidaan luokittelu tehdä formaalien kielten teorian tai graafiteorian menetelmien avulla. Merkkijonojen luokittelussa voidaan käyttää formaaleiden kielten jäsennysmenetelmiä kohteen luokan tunnistamiseen. Tällöin kyseessä on syntaktisten menetelmien perheeseen kuuluva luokitin. Koska virheidentunnistus ja -luokittelu perustuvat oletukseen, että luokiteltavat kohteet pystytään kuvaamaan $d$-ulotteisiksi piirrevektoreiksi, syntaktisten menetelmien käytölle ei tässä työssä ole perusteita.

**Sääntöihin perustuvat menetelmät**. Luokittelu voidaan myös tehdä määrittelemällä joukko luokittelusääntöjä tai johtamalla nämä säännöt opetusaineistosta. Säännöt jakavat piirreavaruuden osiin, jotka muodostavat luokitteluongelman luokat. Käytännössä nämä luokittimet ovat päätöspuita tai vastaavia rakenteita [16, s. 394]. Esimerkkinä tämän tyyppisestä luokittimesta voidaan pitää aliluvussa 3.4 esiteltävää sumeaan päättelyyn perustuvaa luokitinta. Sääntöihin perustuvat menetelmät ovat hyviä silloin, kun laadunvalvonnan luokitteluongelma ei ole selkeästi määritelty, mutta silti mallinnettavissa joukoksi sääntöjä. Sumea päättelykone mahdollistaa asiantuntijan ajattelun mallintamisen, joten sen avulla voidaan yrittää jäljitellä laadunvalvojan päätöksentekoa [30, s. 23-24].

**Malleihin perustuvat menetelmät**. Näissä menetelmissä tietovarastoon on tallennettuna kunkin luokan edustajia eli mallihahmoja (prototyyppejä). Kutakin luokiteltavaa näytettä verrataan mallihahmoihin, jonka jälkeen luokittelupäätös tehdään samankaltaisimpien mallihahmojen perusteella. Eräs tällainen menetelmä on aliluvun 3.3 lähinaapuriluokitin ($k$-Nearest-Neighborhood Classifier), joka vertaamalla piirrevektoria opetusaineistosta muodostettuun kirjastoon päättelee, mihin luokkaan vertailtava kohde kuuluu. Mallihahmoja tarvitaan tyypillisesti menetelmissä paljon. Tämä tekee luokiteltavan kohteen vertailusta mallihahmoihin lasken-

nallisesti raskasta. Menetelmä on tämän vuoksi usein hidas ja muistinkulutukseltaan suuri [1; 51]. Malleihin perustuvat menetelmät sopivat osaan laadunvalvonnan vaikeista luokittelutehtävistä hyvin. Tuotantolinjan jatkeena toimiviin automatisoituihin laadunvalvontapisteisiin nämä menetelmät eivät aina sovellu suorituskykyvaatimustensa takia. Niitä ei siis voida käyttää yleiskäyttöisenä luokittimena konenäkölaadunvalvonnassa.

## 2.5 Tavoitteet ja reunaehdot

Konenäkölaadunvalvonta asettaa löydettyjen kohteiden luokittelulle tiettyjä reunaehtoja. Ne asettavat haasteen luokittelijan opetettavuudelle, erottelukyvylle ja luotettavuudelle. Tässä aliluvussa käydään läpi virheluokittelijaa konenäkölaadunvalvonnan kontekstissa koskevat reunaehdot ja rajoitteet.

Reunaehdoilla on merkitystä varsinkin eri luokittelumenetelmiä eli luokittimia vertailtaessa. Arvioitaessa luokittimen soveltuvuutta konenäkölaadunvalvontaan on vertailtava paitsi luokittelun luotettavuutta, myös opetukseen vaadittavaa aikaa ja vaivaa, opetukseen tarvittavaa näytteiden määrää sekä luokittimen suorituskykyä.

### 2.5.1 Opetettavuus

Virhetyypistä saatavilla olevien näytteiden määrä on usein hyvin rajoitettu. Luokittimen opetusvaiheessa käytettävissä saattaa olla vain muutama virhemalli, jonka avulla uusi virheluokka pitää pystyä määrittelemään. Konenäkölaadunvalvonnassa käytettävään luokittelijaan tulisikin valita menetelmä joka ei vaadi suurta opetusaineistoa.

### 2.5.2 Luotettavuus

Konenäkölaadunvalvonnan tapauksessa luokittelijan erottelukyky ja luotettavuus (eli suorituskyky) riippuu paitsi luokittimesta, myös kuvausjärjestelystä, virheiden piirteiden irrotusmenetelmistä ja kuvannettujen kohteiden koosta bittikartalla. Kuvantamisjärjestelyn ja piirreilmaisimien sopivuus vaadittuun tehtävään on perusedellytys riittävän luokittelutarkkuuden saavuttamiseksi. Mitattavien piirteiden määrän lisääminen ja ennen kaikkea oikea valinta parantavat luotettavuutta. Luokkien lukumäärän lisääntyminen puolestaan heikentää sitä. Tehokkaan ja luotettavan luokittelun edellytyksenä voidaankin pitää luokitteluongelmaan hyvin sopivien mitattavien ominaisuuksien ja luokittelumenetelmien valintaa [9].

Työn tilaajan tyypillisessä konenäkölaadunvalvontatehtävässä luokittelijan tulisi pystyä määrittämään tunnistetulle visuaaliselle virheelle luokka vähintään 90 % - 95 % varmuudella. Kaikki väärät luokittelut eivät kuitenkaan ole yhtä haitallisia. Luokittelijan tulisikin olla sellainen, että sen väärinluokittelukustannusta voidaan

säätää esimerkiksi sakkokertoimilla silloin, kun tietynlaiset väärät luokitukset ovat haitallisempia kuin toiset.

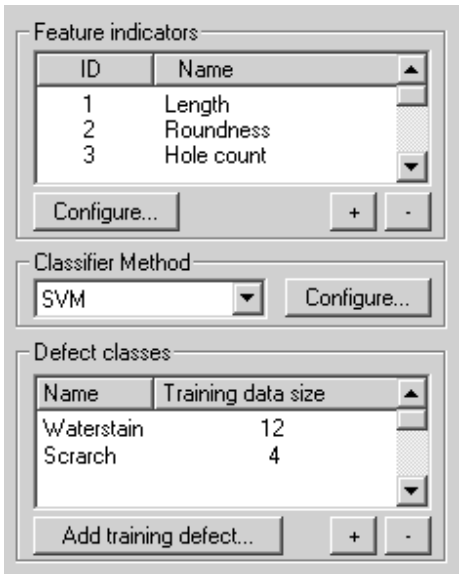
Luokitinmenetelmältä vaaditaan ennen kaikkea robustisuutta. Hyvän menetelmän tulee olla hyvin yleistyvä ja vakaa, eikä se saa olla herkkä opetusaineiston häiriöille. Huber [26, s. 5] listaa robustisen menetelmän ominaisuudet, jotka Łeski [41] on tulkinnut luokittimien tapauksessa seuraavasti:

1. Riittävän hyvä luotettavuus oletetun mallin mukaisissa tapauksissa.
2. Pienet poikkeamat mallin mukaisista oletuksista heikentävät luotettavuutta vain vähän.
3. Suurtenkaan poikkeamien mallin mukaisista oletuksista ei tule aiheuttaa katastrofia.

Tämän työn empiirisessä osassa käytetyt testiaineistot ovat tosimaailmasta, joten ne sisältävät poikkeamia ja häiriöitä. Mahdollisuus robustisuuden arviointiin tuleekin ottaa huomioon testausmenetelmiä valittaessa.

### 2.5.3 Käytettävyys

Luokittelijan on tarkoitus olla mahdollisimman joustava, mutta samalla helposti konfiguroitavissa konenäkösovelluksen käyttöliittymän kautta. Käyttöliittymän avulla asetetaan kulloinkin ratkaistava luokitustehtävä ja sen ratkaisemiseksi käytetyt menetelmät parametreineen.




***Kuva 2.7.*** *Ehdotus luokittelijan käyttöliittymäksi*

Luokittelijan käyttöliittymälle on tiettyjä rajoituksia. Ensinnäkin käyttöliittymän koko voi olla ennalta määrätty. Koko rajoittaa ikkunaan mahtuvien käyttöliittymäkomponenttien määrää. Painikkeiden taakse voidaan kuitenkin aina luoda uusia itsenäisiä asetusikkunoita, joiden avulla voidaan säätää komponentin toimintaa. Toiseksi käyttöliittymän kieli ja sovellusalueen termit on otettava huomioon käyttöliittymää suunniteltaessa. Myös mahdollinen monikielisyystuki asettaa omat haasteensa. Kolmanneksi kaikki konenäköohjelmiston käyttäjät eivät ole konenäön ammattilaisia. Siksi käsitteiden ja käyttöliittymän tulisikin olla ymmärrettäviä ja käytettäviä myös maallikoille. Neljänneksi käyttöliittymän suunnittelussa tulee käyttää ohjelmistoalustan käyttöliittymäohjeistusta. Windows-sovellusten tapauksessa tämä on Microsoftin Windows-käyttöliittymien suunnitteluun tarkoitettu ohje [46].

Kuvassa 2.7 on esitelty luonnos luokittelijan käyttöliittymästä. Piirreilmaisimien, luokittelumenetelmän, virheluokkien ja opetusaineiston valinta mahdollistaa sopivimman yhdistelmän löytämisen käsillä olevan luokitteluongelman ratkaisemiseksi. Käyttöliittymästä tulee lisäksi päästä muuttamaan valitun luokittimen parametreja, sillä parametrien säätäminen saattaa olla välttämätöntä luokittimen luokitteluvarmuuden parantamiseksi. Parametrien säätö voidaan toteuttaa esimerkiksi erillisen ikkunan avulla.

### 2.5.4 Nopeus

Erityisesti tuotantoprosessin yhteydessä olevan niin sanotun on-line -konenäköjärjestelmän tarkistusnopeudelle asetetaan käytännön sovelluksissa tiukkoja reunaehtoja. Konenäön ei saa hidastaa tuotantoprosessia, vaan sen pitää olla riittävän nopea toimiakseen prosessin osana. Tuotantoprosessi toimii tyypillisesti sykleissä, joiden aikana valmistuu tarkastettavaksi muutamia tuotteita riippuen esimerkiksi ruiskuvalukoneen muottipesien määrästä.

Reaaliaikaisuuden tulee Brzakovicin [4] mukaan olla yksi laadunvalvontajärjestelmän tavoitteista. Tässä työssä käytetään Brzakovicin määritelmää reaaliaikaisuudesta, jonka mukaan järjestelmä on reaaliaikainen, jos se pysyy tiedon hankinnan ja tuotannon vauhdissa. Brzakovicin reaaliaikaisuus ja on-line -laadunvalvonta tarkoittavat siis samaa asiaa.

Luokittimen suoritusaika on suoraan verrannollinen luokiteltavien kohteiden määrään. Suoritusaikaan vaikuttaa suuresti luokittimen käyttötapa tarkastussyklin aikana. Jos luokittelijaa käytetään apuna hylkäyspäätöstä tehtäessä, on luokittelijan käyttötarkoitus pääasiassa tarkastustarkkuuden parantaminen. Tällöin yhden tarkastussyklin aikana luokiteltavana saattaa olla joitain satoja näytteistä segmentoituja virheitä. Jos taas luokittelijaa käytetään laatutiedon keräämisen tuotantoprosessin tilan seuraamiseksi, luokiteltavia kohteita on yhden syklin aikana korkeintaan joitain kymmeniä.

Suoritusajalle asetettavat reunaehdot vaihtelevat sovelluskohtaisesti. Taulukossa 2.1 on tyypillisiä konenäkötarkastuksen suoritusaikoja. Taulukossa **typ.** tarkoittaa tyypillistä jaksonaikaa, **maks.** pisintä sallittua jaksonaikaa ja **marg.** tarkoittaa luokittelijan käytettävissä olevaa aikamarginaalia. Suorituskyvyn kannalta haasteellisimpia ovat massatuotannon laatua valvovat sovellukset.

***Taulukko 2.1.*** *Konenäkötarkastuksen suoritusaikoja*

| **Sovellus** | **typ.** | **maks.** | **marg.** |
|---|---|---|---|
| Kännykän näytön lasin tarkastus | 5-7 s | 8 s | 3-1 s |
| Maalattujen kappaleiden tarkastus | 3 s | 4 s | 1 s |
| Mikromaailma, ei-reaaliaikainen mittaus | 5 s | 10+ s | 5+ s |

Vaikka luokittelija antaa laadunvalvonnalle selkeää lisäarvoa ja parantaa sen tarkkuutta, tulee luokittelun silti säilyttää on-line -tarkastuslaitteistojen reaaliaikaisuus. Työssä tutkitaan myöhemmin valittavien ja esiteltävien luokitinmenetelmien kykyä selvitä taulukon 2.1 mukaisista suoritusaikavaatimuksista niin tarkastavassa (korkeintaan 100 luokittelua / sykli) kuin tietoa keräävässä (korkeintaan 10 luokittelua / sykli) roolissa.

## 2.6 Luokittimien valinta

Tämän työn onnistumisen edellytyksenä on valita erilaisista kirjallisuudesta löytyvistä luokittelumenetelmistä toteutettavaksi luokittimet, jotka parhaiten soveltuvat käytettäväksi konenäkölaadunvalvonnassa. Luvun 2.5 tavoitteiden ja reunaehtojen lisäksi valintaan vaikuttaa työn tekijän aikaisempi kokemus sumeasta logiikasta. Myös luokittimen kompleksisuus vaikuttaa sen todennäköisyyteen tulla valituksi. Tekijän kokemus ohjelmistoalalta on opettanut, että yksinkertaiset ratkaisumallit tekevät projektin onnistumisesta todennäköisemmän (KISS=”Keep it Simple, Stupid” periaate).

### Vaihtoehdot ja valintakriteerit

Työn edetessä tuli selväksi, että työssä voidaan toteuttaa vain valittu joukko erilaisia menetelmiä. Hyödyllisiä lähteitä luokittimien valinnassa olivat Duda et al. *Pattern Classification* [16] ja Webb [70], joissa esitellään erilaisia luokittelumenetelmäperheitä ja analysoidaan niiden soveltuvuutta erilaisiin luokittelutehtäviin, sekä valmistusteollisuuden konenäkökatsaukset, kuten Malamas et al. *A survey on industrial vision systems, applications and tools* [44].

Valintaa helpottamaan määriteltiin visuaalisten virheiden luokittelijan valintakriteerit. Ensinnäkin luokittimen oppimistavan ja laskennallisten menetelmien tuli

soveltua hyvin virheiden luokitteluun. Toiseksi luokittimien tuli olla riittävän yksinkertaisia toteutettavaksi ja ylläpidettäväksi osana laadunvalvontaohjelmistoa. Kolmas vaatimus oli, että valittujen luokittimien tuli toteuttaa luvussa 2.5 määritellyt rajoitteet ja reunaehdot. Valintaan vaikutti myös tekijän aikaisempi kokemus useimpien valituksi tulleiden luokittimien matemaattisista perusteista.

Nämä kriteerit täyttäviä menetelmiä variantteineen olisi silti ollut kymmeniä, joten viimeiseksi valintakriteeriksi valikoitui menetelmän yleisyys kirjallisuudessa. Kirjallisuudesta löytyi useita artikkeleita, joissa luokitteluongelmaa oli lähestytty konenäön näkökulmasta. Harvassa tutkimuksessa oli kuitenkaan yritetty ratkaista konenäön luokitteluongelmaa käyttäen useampaa kuin yhtä menetelmää, joten sovelluskohtaisista artikkeleista voitiin ainoastaan tehdä huomioita luokittelumenetelmien suosiosta. Menetelmät, jotka olivat usein edustettuina tapaustutkimuksissa ja luokitinmenetelmiä vertailevissa tutkimuksessa [45; 23; 17; 58; 40; 4; 54] olivat todennäköisimpiä toteutettavaksi harkittavia.

### Valitut luokittimet

Tutkittavaksi päätettiin ottaa neljä erityyppistä luokittelumenetelmää. Yksinkertaisin työssä toteutettu luokitin oli mallihahmojen vertailuun perustuva $k$-NN lähinaapuriluokitin (*$k$-Nearest-Neighbour classifier*), joka toimi tässä työssä referenssiluokittimena, jonka luokittelukykyä, robustisuutta ja suorituskykyä vasten muita luokittimia vertailtiin. Tilastollisista luokittimista toteutettavaksi valittiin naiivi Bayes -luokitin, sillä se on luokitteluongelmaa yksinkertaistavista oletuksistaan huolimatta menestynyt käytännön luokittelutehtävistä yllättävän hyvin [59]. Monimutkaisempia luokittelutehtäviä varten arvioitavana oli SVM-luokitin (*Support Vector Machine*, eli tukivektorikone). Lisäksi vaikeasti mallinnettavia ja määriteltäviä luokitteluongelmia varten toteutettavaksi valittiin sumean säädön teoriaan perustuva luokitin (*Fuzzy Control Classifier*).

Valitut luokittelumenetelmät ovat tarkoituksella toimintaperiaatteiltaan ja ominaisuuksiltaan erilaisia. Niiden eroja on pyritty kartoittamaan taulukkoon 2.2. Taulukon edut ja haitat on koottu kirjallisuudesta. k-NN ja naiivin Bayes -luokittimien osalta lähteenä on Duda et al. 2001 [16], SVM-luokittimen kohdalla Cristianini & Shawe-Taylor 2000 [13] ja sumean päättelykoneen kohdalla Ross 1995[60].

### Hylätyt vaihtoehdot

Laadunvalvontaan soveltuva luokitin olisi mahdollinen toteuttaa myös käyttäen neuroverkkoja, Parzen-ikkunoita, päätöspuita, tiedonlouhinnassa käytettäviä klusterointimenetelmiä tai lukuisia muita matemaattisia menetelmiä. [16] Erilaisia menetelmiä on testattu kattavasti STATLOG-projektissa, jossa testattiin 22 eri luokitinmene-

***Taulukko 2.2.*** *Valittujen luokittelumenetelmien ominaisuuksien vertailua. Taulukossa luokittimet on nimetty seuraavasti; k-NN = lähinaapuriluokitin, NB = naiivi Bayes -luokitin, SVM = tukivektorikoneluokitin ja FCC = sumea luokitin.*

| Luokitin | Edut | Haitat |
|---|---|---|
| **k-NN** | + Yksinkertainen | - Laskennallisesti raskas<br>- Hyvin riippuvainen opetusaineiston kattavuudesta |
| **NB** | + Yksinkertainen<br>+ Suorituskykyinen | - Riippumattomuusoletus<br>- Virhejakaumat tunnettava tai laskettava. |
| **SVM** | + Suoriutuu epälineaarisista luokittelutehtävistä<br>+ Ilmaisuvoimainen | - Laskennallisesti raskas |
| **FCC** | + Opetusaineisto ei välttämätöntä<br>+ Yksinkertainen toteuttaa<br>+ Läpinäkyvyys | - Aineistosta oppivuus.<br>- Suorituskyky |

telmää 22:lla eri testiaineistolla. Erilaisista luokittelumenetelmistä ja niiden eroista kiinnostuneiden kannattaakin tutustua projektin raporttiin [45]. Tuoreempi vertailu on Caruanan ja Niculescu-Mizilin *An empirical comparison of supervised learning algorithms* [6], jossa he vertailevat STATLOGin jälkeen kehitettyjä uusia menetelmiä. Osana esitutkimusta päädyttiin kuitenkin rajaamaan tutkittavat menetelmät luvussa 3.2 esiteltäviin luokitinvaihtoehtoihin.

# 3. MATEMAATTINEN TEORIA

Tämä luku käsittelee luokittelussa käytettyjen matemaattisten menetelmien teoriaa konenäköpohjaisen laadunvalvonnan kontekstissa. Teoriaosaan ei sisälly kuvausta niistä menetelmistä, joilla luokiteltavat kohteet löydetään ja irrotetaan bittikartoista. Tämän tekniikan oletetaan olevan käytettävissä luokittelijaa varten.

## 3.1 Luokittimen formalisointi

Tämä työ keskittyy tilastollisiin luokittimiin, joissa luokittelu tehdään piirrevektoreiden avulla. Tällöin kullekin luokiteltavalle kohteelle tulee piirreilmaisimien avulla laskea piirrevektori $\mathbf{x} \in X$. Tässä työssä piirreavaruus $X = \mathbb{R}^d$.

Merkitään luokkien joukkoa notaatiolla $\Omega = \{\omega_j \ : \ j = 1, \ldots, q\}$. Tässä $q$ on luokitteluongelman luokkien lukumäärä. Usein on käytännöllisempää viitata luokkiin niiden tunnisteilla $Y = \{1, 2, \ldots, q\}$, joista kullakin on vastineluokka siten, että kaikille $\omega_j$ on olemassa yksikäsitteinen $j$ siten, että $j \in Y$.

Kaikki tilastolliset luokittimet toimivat jakamalla piirreavaruuden $X$ luokkia $\omega_j$ vastaaviin pistevieraisiin osajoukkoihin $A_j$. Näitä osajoukkoja sanotaan *päätösalueiksi* ja alueita erottavia pintoja *päätöspinnoiksi* [16, s. 30].

**Määritelmä 1** *Luokittelija on funktio $c : X \to Y$, joka kuvaa piirrevektorin $\mathbf{x} \in X$ luokkatunnisteeksi $y \in Y$.*

Luokittimen määritelmästä seuraa, että päätösalueet voidaan määritellä seuraavasti [16, s. 30]:

$$A_j = \{\mathbf{x} \in X : c(\mathbf{x}) = j\} \tag{3.1}$$

Päätösfunktio $g_j : \mathbb{R}^d \to \mathbb{R} \quad \forall j \in Y$ kertoo, kuinka edullista on luokitella näyte piirrevektorin $\mathbf{x}$ perusteella luokkaan, jonka tunniste on $j$. Jos kaikille luokille $j \in Y$ on määritelty päätösfunktiot $g_j$, voidaan luokittelija esittää niiden avulla seuraavasti:

$$c(\mathbf{x}) = \arg \max_{1 \leq j \leq q} (g_j(\mathbf{x})), \tag{3.2}$$

missä arg max on funktio, joka palauttaa sen argumentin $j$ arvon, jolla päätösfunktiot $g_j$ palauttavat suurimman arvonsa. Luokituksen tulee aina olla yksikäsitteinen,

joten määritellään yhtäsuurien päätösfunktioiden arvojen tapauksessa palautettavaksi pienimmän luokkatunnisteen $j$ arvo.

Luokittimen opettamisessa käytetään seuraavaa opetusjoukon määritelmää.

**Määritelmä 2** *Joukkoa $D$, jossa on $l$ kappaletta piirrevektori-luokkatunnisteparia $(\boldsymbol{x}_i, y_i), \boldsymbol{x}_i \in \mathbb{R}^d$ ja $y_i \in Y$ sanotaan opetusjoukoksi.*

Luokittimet opetetaan siten, että opetusjoukon $D$ avulla rakennetaan päätösfunktiot $g_j$ kaikille luokkatunnuksille $j \in Y$, minkä jälkeen luokitinfunktio saadaan yhdistämällä päätösfunktiot kaavan (3.2) mukaisesti. Päätösfunktioiden muoto ja käytetyt opetusmenetelmät vaihtelevat luokitinmenetelmäkohtaisesti.

## 3.2 Naiivi Bayes-luokitin

Ihminen käyttää luokitellessaan aiempaa tietämystään ilmiöistä ja niiden riippuvuuksista. Esimerkiksi laadunvalvontaa suorittava henkilö on oppinut, että pitkänomainen epäsäännöllisyys tarkastettavassa pinnassa on todennäköisimmin naarmu. Bayes-luokittelussa tämä tosimaailman syy-seuraus -suhteisiin perustuva päättely mallinnetaan matemaattiseksi malliksi, jossa luokittelu tehdään virhetyyppien suhteellisten esiintymistodennäköisyyksien ja tyyppiin liittyvien piirrejakaumien välisen yhteyden avulla [16]. Tilastollisia yhteyksiä (prioritodennäköisyyksiä) päättelyssä käyttävät Bayes-luokittimet ovat robusteja, sillä niiden olemukseen sisältyy ilmiöiden esiintymisen ja käytettävissä olevan tiedon epävarmuus. Tämä on tarpeen, sillä kaikkeen luokitteluun kuuluu aina pieni määrä epävarmuutta [39].

Jo tapahtuneen tapahtuman $A$ perusteella voidaan arvioida tapahtuman $B$ todennäköisyyttä, jos näiden tapahtumien välinen tilastollinen riippuvuus (ehdollinen todennäköisyys) tunnetaan. Bayesin teoreema (1) mallintaa tätä riippuvuutta.

**Teoreema 1** *Tapahtuman $B$ todennäköisyys ehdolla $A$ on ja tapahtuman $A$ todennäköisyys ehdolla $B$ välillä on suhde, jota voidaan kuvata seuraavasti:*

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Bayesin teoreeman lisäksi Bayes-luokittimen määrittelemiseksi tulee ensin esitellä muutamia käsitteitä, kuten satunnaismuuttujavektori $\mathbf{X} = (X_1, ..., X_d)$, jonka kukin satunnaismuuttujajäsen $X$ edustaa yhtä mitattavaa piirrettä. Otetaan lisäksi $Y$, joka on luokkatunnistetta edustava satunnaismuuttuja, joka saa arvoja joukosta $\{1, \ldots, q\}$. Satunnaismuuttujia merkitään isoilla kirjaimilla, kuten $X_i$, kun taas pienillä kirjaimilla kuten $x_i$ merkitään satunnaismuuttujan saamia arvoja. Bayes-luokitin rakennetaan käyttäen annetulle piirrevektorille laskettavia seuraavaksi johdettavia päätösfunktioita. Luokittelussa haluamme estimoida luokituksen todennäköisyyksiä, kun pirrevektori tunnetaan:

$$P(Y = j \mid \mathbf{X} = \mathbf{x}) \tag{3.3}$$

Soveltamalla kaavaan (3.3) Bayesin teoreemaa, saadaan

$$\frac{P(\mathbf{X} = \mathbf{x} \mid Y = j)\ P(Y = j)}{P(\mathbf{X} = \mathbf{x})}. \tag{3.4}$$

Kaavan mukaan todennäköisyys sille, että kohde, jota piirrevektori $\mathbf{x}$ kuvaa, kuuluu luokkaan $\omega_j$ saadaan kun kerrotaan piirrevektorin $\mathbf{x}$ esiintymisen todennäköisyys luokassa $\omega_j$ luokan esiintymistodennäköisyydellä koko aineistossa ja jaetaan se piirrevektorin $\mathbf{x}$ esiintymistodennäköisyydellä.

Koska $P(\mathbf{X} = \mathbf{x})$ on kaikille luokkakohtaisille posterioritodennäköisyyksille sama, se voidaan poistaa nimittäjästä, jolloin saadaan Bayes-luokittimen päätösfunktio:

$$g_j^*(\mathbf{x}) = P(\mathbf{X} = \mathbf{x} \mid Y = j)\ P(Y = j) \tag{3.5}$$

Johdetun päätösfunktion määritelmän avulla voidaan määritellä Bayes-luokitin käyttäen luvun 3.1 notaatiota:

$$c_{\mathrm{BC}}(\mathbf{x}) = \arg\max_{1 \le j \le q} \{P(\mathbf{X} = \mathbf{x} \mid Y = j)\ P(Y = j)\} \tag{3.6}$$

Bayes-luokitin luokittelee piirrevektorin $\mathbf{x}$ edustaman kohteen kuuluvaksi siihen luokkaan, jonka posterioritodennäköisyys on suurin (MAP, *maximum a posteriori probability*). Koska piirrekohtainen todennäköisyys riippuu myös muiden piirteiden saamista arvoista on todennäköisyyden $P(\mathbf{X} = \mathbf{x} \mid Y = j)$ arvioiminen vaikeaa varsinkin piirreavaruuden dimension kasvaessa. Tämän vuoksi usein käytetäänkin yksinkertaistavaa oletusta, että piirteiden tilastolliset muuttujat ovat toisistaan riippumattomia.

$$P(Y = j \mid \mathbf{X} = \mathbf{x}) = \frac{P(\mathbf{X} = \mathbf{x} \mid Y = j)P(Y = j)}{P(\mathbf{X} = \mathbf{x})} \tag{3.7}$$

$$\propto P(X_1 = x_1, \ldots, X_d = x_d \mid Y = j)P(Y = j) \tag{3.8}$$

$$= P(Y = j) \prod_{i=1}^{d} P(X_i = x_i \mid Y = j) \tag{3.9}$$

Tämän oletuksen avulla rakennettua luokitinta (3.10) sanotaan naiiviksi Bayes-luokittimeksi.

$$c_{\mathrm{NBC}}(\mathbf{x}) = \arg\max_{1 \le j \le q} P(Y = j) \prod_{i=1}^{d} P(X_i = x_i \mid Y = j) \tag{3.10}$$

Huolimatta tästä epärealistisesta oletuksesta naiivi Bayes -luokitin on osoittautunut käytännössä menestykselliseksi luokittelumenetelmäksi [59]. Riippumattomuusoletus on myös monissa tapauksissa perusteltua, kuten Hand ja Yu toteavat artikkelissaan *Idiot's Bayes* [22]. Virheiden luokittelussa riippumattomuusoletus on validi, kunhan piirteiden arvot eivät riipu toisistaan [59].

Luokkien todennäköisyystiheysjakaumat $p(x_i \mid \omega_j)$ piirteiden $X_i$ funktioina voidaan estimoida opetusaineistosta $D$ tai ne voidaan syöttää käsin, mikäli opetusdataa ei ole riittävästi saatavilla. Usein esimerkiksi virhetyyppien esiintymistodennäköisyydet $P(Y = j)$ tunnetaan, joten ne voidaan syöttää malliin sellaisenaan.

Koska Bayes-luokittimen toiminta perustuu prioritodennäköisyysjakaumiin, on niiden tuottaminen opetusdatasta hyvin tärkeässä roolissa. Bouckaert [3] on testannut naiivin Bayes-luokittimen rakentamiseen hyvin toimivia todennäköisyysjakauman estimointimenetelmiä. Seuraavaksi esitellään kolme erilaista prioritodennäköisyysjakaumien estimointimenetelmää Bouckaertiä mukaillen.

Yksinkertaisin ratkaisu piirrejakaumien estimointiin on olettaa opetusaineiston piirteiden satunnaismuuttujat $X_i$ luokkakohtaisesti normaalijakautuneiksi:

$$p(x_i \mid \omega_j) \sim N(\mu, \sigma^2) \tag{3.11}$$

Koska tiedämme, että normaalijakauman tiheysfunktio on muotoa

$$f_{\mathrm{NDF}}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{3.12}$$

voidaan jakauman keskiarvo $\mu$ ja keskihajonta $\sigma$ laskea valitsemalla opetusjoukosta $D$ luokkakohtaiset piirrearvot. Tätä varten tarkastellaan opetusaineistoa siten, että se on järjestetty luokkia $j \in Y$ vastaaviin ryhmiin.

$$D_j = \{(\mathbf{x}, y) \mid (\mathbf{x}, y) \in D, y = j\} \tag{3.13}$$

Nyt luokkakohtaiset keskiarvot ja keskihajonnat voidaan laskea.

$$\mu_{j_i} = \frac{1}{\mid D_j \mid} \sum \{x_i \mid (\mathbf{x}, y) \in D_j\} \tag{3.14}$$

$$\sigma_{j_i} = \sqrt{\frac{1}{\mid D_j \mid} \sum \{(x_i - \mu_{j_i})^2 \mid (\mathbf{x}, y) \in D_j\}} \tag{3.15}$$

Jakaumat $p(x_i \mid \omega_j)$ voidaan lopulta ilmaista seuraavasti:

$$p(x_i \mid \omega_j) = \frac{1}{\sigma_{j_i}\sqrt{2\pi}} \exp\left(-\frac{(x - \mu_{j_i})^2}{2{\sigma_{j_i}}^2}\right) \tag{3.16}$$

Koska piirrekohtaisten satunnaismuuttujien $X_i$ jakaumat eivät useinkaan ole nor-

maalisti jakautuneita, ei tähän oletukseen perustuva estimointimenetelmä tuota aina luotettavaa luokitinta.

Ei-normaalijakautuneita priorijakaumia voidaan mallintaa muodostamalla jakaumat summaamalla ydinfunktioita $K$ [3]. Summattavien ydinfunktioiden paikat ovat tyypillisesti määritelty opetusjoukon $D$ piirrevektoreiden $\mathbf{x}_k$ avulla [53].

$$p(x_i|\omega_j) = \frac{1}{lh} \sum_{k=1}^{l} K\left(\frac{x_i - x_{ki}}{h}\right), \tag{3.17}$$

missä $l$ on opetusjoukon koko ja samalla jakauman tiheysfunktion estimointiin käytettyjen ydinfunktioiden lukumäärä, $h$ on sopivasti valittu ydinmenetelmän ikkunan leveys ja $x_{ki}$ opetusjoukon $D$ $k$:nnen piirrevektorin $i$:s alkio.

Bouckaert [3] kuvaa pintapuolisesti tämän menetelmän, mutta kuvaus on osin epätarkka ja virheellinenkin, sillä Bourckaert puhuu varianssista, kun tarkoitetaan keskihajontaa. Menetelmälle parempi lähde on Johnin ja Langleyn *Estimating continuous distributions in bayesian classifiers* [33]. Normaalijakaumaoletuksen sijaan John ja Langley summaavat normaalijakauman tiheysfunktioiden muotoisia ytimiä, joiden massakeskipisteet ovat opetusjoukon piirrearvojen kohdilla. Tämä tiheysfunktioestimaatti voidaan esittää kaavana

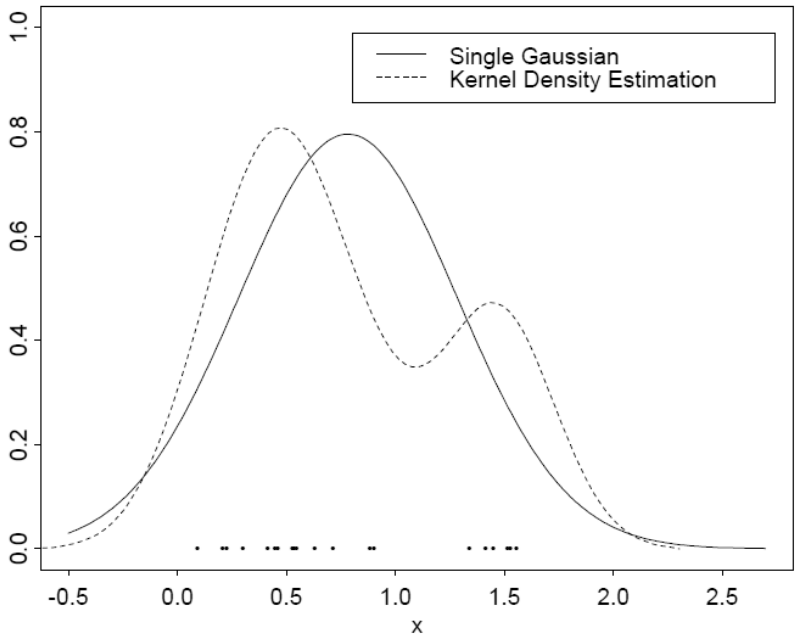
$$p(x_i|\omega_j) = \frac{1}{\mid D_j \mid} \sum_{k=1}^{|D_j|} f_{\mathrm{NDF}}(x_i; \mu_k, \sigma_j), \tag{3.18}$$

missä keskiarvo $\mu_k$ voidaan korvata opetusjoukon $D_j$ piirrevektorin $\mathbf{x}_k$ piirteen $x_{ki}$ arvolla. Tiheysfunktion keskihajontana John ja Langley käyttävät arvoa $\sigma_j = 1/\sqrt{\mid D_j \mid}$, missä $\mid D_j \mid$ on luokituksen $j$ omaavien näytteiden määrä opetusjoukossa $D$.

John ja Langley [33] ovat analysoineet estimaatin tarkentuvuutta ja todenneet sen olevan vahvasti pisteittäin tarkentuva (strongly pointwise consistent) jakauman $p(j|x)$ estimaatti. Näin ollen opetusjoukon koon kasvaessa ydinmenetelmän estimoidut jakaumat lähestyvät asymptoottisesti optimaalisen Bayes-luokittimen määrittäviä jakaumia.

Kuvassa 3.1 on havainnollistettu, miten ydinmenetelmä seurailee pisteiden jakautumista normaalijakaumaa paremmin, mikäli pistejoukon arvot eivät ole normaalijakautuneet. Haittapuolena tässä menetelmässä on sen aiheuttama laskentakompleksisuuden kasvu. Siinä missä normaalijakautuneiksi oletettujen prioritodennäköisyyksien $P(\mathbf{x} \mid C)$ laskemiseksi riittää, kun evaluoidaan normaalijakauman tiheysfunktio $f(x_i)$ $q \times d$ kertaa, joudutaan ydinmenetelmässä evaluointi tekemään $q \times d \times l$ kertaa. Tässä $l$ on opetusdatan näytteiden lukumäärä.

Kolmas Bouckaertin testaama ja suosittelema jakaumatyyppi on diskretisoitu jakauma. Siinä jatkuva piirteiden satunnaismuuttujaa $X$ estimoidaan diskreetillä satunnaismuuttujalla $X'$, jonka jälkeen jakaumat $P(X' \mid C)$ estimoidaan klassisen




***Kuva 3.1.*** *Normaalijakauman ja ydinmenetelmän ero jatkuvan satunnaismuuttujan tiheysfunktion estimoinnissa [33]*

todennäköisyysjakauman avulla seuraavasti

$$p(x' \mid \omega_j) = \frac{|D_{x'j}|}{|D_j|}, \tag{3.19}$$

missä $|D_{x'j}|$ on niiden opetusjoukon alkioiden lukumäärä, joiden luokitus on $j$ ja joiden diskretisoitu arvo on $x'$ ja $|D_j|$ vastaavasti luokkaan $j$ luokiteltujen alkioiden lukumäärä.

Satunnaismuuttujien $X$ diskretisointiin Bouckaert suosittelee Fayyadin ja Iranin [19] MDL (*minimum description length*) -menetelmää. Menetelmä perustuu heuristiikkaan, joka rekursiivisesti valitsee pienimmän entropiamuutoksen aiheuttavan leikkauksen diskretisoinnin rajaksi.

Joukon $S$ luokkaentropia lasketaan kaavalla

$$Ent(S) = -\sum_{i=1}^{k} P(\omega_i, S) \log(P(\omega_i, S)), \tag{3.20}$$

missä $k$ on luokkien lukumäärä ja $P(\omega_i, S)$ luokituksen $\omega_i$ omaavien näytteiden suhde joukossa $S$.

Diskretisointi tehdään piirrekohtaisesti. Merkitään diskretisoitavaa piirrettä kirjaimella $A$. Jos oletetaan, että $l$ näytteisen opetusjoukon $D_A$ piirrearvot $x_i$ ovat suuruusjärjestyksessä, eli $x_a < x_b$ aina kun $a < b$, niin leikkauskohtakandidaatteja $t \in T_A$ ovat vierekkäisten opetusjoukon $D_A$ pisteiden keskikohdat. Formaalisti leikkauskandidaatit voidaan määritellä seuraavasti:

$$T_A = \left\{ \frac{x_i + x_{i+1}}{2} : i = 1 \dots l-1 \right\} \tag{3.21}$$

Leikkauskohdan $t \in T_A$ indusoima $S$ luokkaentropian muutos voidaan laskea kaavalla

$$Gain(A, t; S) = Ent(S) - \frac{|S_1|}{|S|} Ent(S_1) - \frac{|S_2|}{|S|} Ent(S_2), \tag{3.22}$$

missä $S_1 \subset S : x < t$ ja $S_2 \subset S : x > t$ ja $|S|$ on joukon $S$ alkioiden lukumäärä.

Leikkauskandidaateista valitaan se, jolle $Gain(A, t; S)$ on suurin.

$$t_{best} = \arg\max_{t \in T} \left( Gain(A, t; S) \right) \tag{3.23}$$

Fayyad ja Irani [19] käyttävät leikkauskandidaatin hyväksymiseen MDLP -koodausteoriasta johdettua ehtoa, joka on muotoiltu teoreemassa 2.

**Teoreema 2** *Leikkauskohtakandidaatti $t_{best}$ on leikkauskohta jos*

$$Gain(A, t_{best}; S) > \frac{\log_2(N-1)}{N} + \frac{\Delta(A, t; S)}{N}, \text{ missä}$$

$$\Delta(A, t; S) = \log_2(3^k) - kEnt(S) + k_1 Ent(S_1) + k_2 Ent(S_2),$$

*missä $k$, $k_2$ ja $k_1 \leq q$ ovat joukko- ja osajoukkokohtaiset luokkien määrät eli montako eri luokkaa joukossa/osajoukossa esiintyy.*

Opetusjoukon $D_A$ piirrearvojoukkoa leikataan rekursiivisesti etsimällä aina osajoukkojen parhain ehdokas kaavalla (3.23), kunnes teoreeman 2 leikkausehto hylkää osajoukon parhaan leikkausehdokkaan. Jos leikkauskandidaatti $t$ täyttää leikkausehdon, lisätään se hyväksyttyjen leikkauspisteiden joukkoon $R_A \subseteq T_A$.

Leikkauspisteiden etsinnän jälkeen tiheysfunktion diskreetit arvot lasketaan leikkauskohtien muodostamille arvoväleille. Oletetaan, että hyväksyttyjen leikkauspisteiden joukon $R_A$ alkiot ovat nousevassa järjestyksessä. Ensimmäinen arvoväli on aina $(-\infty, r_1)$ ja viimeinen $(r_m, \infty)$, kun $m$ on hyväksyttyjen leikkauskohtien lukumäärä. Yhteensä diskreettejä kategorioita muodostuu näin ollen $m + 1$ kappaletta. Kategorioiden arvot saadaan laskemalla luokkakohtaiset osuudet kussakin kategoriassa kaavan (3.19) mukaisesti.

Yllä kuvatun diskretisoinnin voi ajatella olevan luokitin luokittimen sisällä. Diskretisoituja jakaumia voitaisiin nimittäin käyttää myös päätöspuita rakennettaessa. Käytettäessä diskretisoituja jakaumia naiivin Bayes -luokittimen kanssa pystytään hyödyntämään Bayes-luokittimen etuja, kuten helposti luokkaprioritodennäköisyyksien kautta säädettävää luokittelutulosten painotusta.

Yllä ehdotetuilla jakaumienestimointimalleilla pyritään parantamaan naiivin Bayes -luokittimen ennustuskykyä. Estimointimallien luokittelutarkkuuksissa, luokittelusuorituskyvyssä sekä opetus- ja suoritusaikavaatimuksissa on eroja, joten niistä voidaan valita kulloinkin ratkaistavana olevaan luokitteluongelmaan sopivin.

## 3.3 Lähinaapuriluokitin

Lähinaapuriluokittimen ($k$-Nearest-Neighborhood classifier, $k$-NN) toimintaperiaate on yksinkertainen. Luokitinta rakennettaessa valitaan $k$, joka on pieni pariton nollaa suurempi kokonaisluku (esimerkiksi 3). Luokiteltavalle pisteelle etsitään opetusaineistosta $k$ lähintä naapuria. Tässä metriikkana voidaan käyttää esimerkiksi euklidista etäisyyttä piirrevektoriavaruudessa. Tämän jälkeen tutkitaan, minkä luokan jäsenet esiintyvät $k$:n lähimmän naapurin keskuudessa eniten ja merkitään luokiteltava kohde kuulumaan kyseiseen luokkaan. [16, s. 183].

Lähinaapuriluokittimen rajoitteena on se, että se on ohjatun oppimisen menetelmä. Luokkien määrä tulee tuntea ennalta ja opettajan tulee voida antaa jokaiselle opetusaineiston alkioille luokka, johon se kuuluu. Lähinaapuriluokittimen opettaminen malleihin perustuvana ei-parametrisena luokittimena ei vaadi luokkien ja piirteiden esiintymistodennäköisyysjakaumien tuntemusta, sillä opetusaineiston etukäteisluokittelu riittää [51].

Triviaali toteutus $k$-NN luokittimelle on laskea luokiteltavan näytteen piirrevektorin $\mathbf{x} \in \mathbb{R}^d$ etäisyys kaikkiin opetusjoukon $D$ piirrevektoreihin, jonka jälkeen tiedettäisiin, mitkä ovat $k$ lähintä opetusjoukon pistettä. Tämä on kuitenkin piirreavaruuden dimension ja opetusjoukon koon kasvaessa laskennallisesti erittäin hidas operaatio. Kirjallisuudesta löytyykin menetelmiä, joilla lähimpien naapureiden löytämistä voidaan tehostaa huomattavasti. Ennen näiden menetelmien esittelyä käydään läpi muutamia merkintöjä.

Merkitään kahden vektorin $\mathbf{a}$ ja $\mathbf{b}$ etäisyyttä seuraavasti:

$$\mathrm{dist}(\mathbf{a}, \mathbf{b}) = f_d(\mathbf{a}, \mathbf{b}) \tag{3.24}$$

Tässä $f_d$ on etäisyysfunktio, jonka tulee täyttää seuraavat metriikan määritelmän mukaiset ominaisuudet:

$$\mathrm{dist}(\mathbf{a}, \mathbf{b}) \geq 0 \tag{3.25}$$

$$\mathrm{dist}(\mathbf{a}, \mathbf{b}) = 0 \quad \text{joss} \quad \mathbf{a} = \mathbf{b} \tag{3.26}$$

$$\mathrm{dist}(\mathbf{a}, \mathbf{b}) + \mathrm{dist}(\mathbf{b}, \mathbf{c}) \geq \mathrm{dist}(\mathbf{a}, \mathbf{c}) \tag{3.27}$$

Etäisyysfunktiona voidaan käyttää mitä tahansa metriikkaa, sillä määritelmänsä mukaisesti ne täyttävät edellä esitellyt vaatimukset. Yleisimmin käytettyjä metriikoita ovat Minkowski-etäisyys (3.28) ja euklidinen etäisyys (3.29).

$$L_m(\mathbf{a}, \mathbf{b}) = \left( \sum_{i=1}^{d} \|a_i - b_i\|^m \right)^{1/m} \tag{3.28}$$

$$L_2(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2} \tag{3.29}$$

Oletetaan opetusjoukko $D$, jossa on $l$ kappaletta piirrevektori- luokkatunnisteparia. Koistinen [36] esittää $k$-NN ($1 \leq k \leq l$) luokitteluongelman seuraavasti. Kun $\mathbf{x}$ on kiinnitetty, niin etsitään sellainen opetusjoukon tunnisteiden $1, \ldots, l$ permutaatio $i_1, \ldots, i_l$, että $\text{dist}(\mathbf{x}, \mathbf{x}_{i1}) \leq \text{dist}(\mathbf{x}, \mathbf{x}_{i2}) \leq \ldots \leq \text{dist}(\mathbf{x}, \mathbf{x}_{il})$

Kaikille luokille $j \in 1 \ldots q$ voidaan laskea, kuinka moni $k$:sta lähimmästä opetusjoukon piirrevektorista $\mathbf{x}_{i1}, \ldots, \mathbf{x}_{ik}$ on peräisin luokasta $j$.

$$k_j = |\{p : y_{ip} = j, p = 1, \ldots, k\}| \tag{3.30}$$

Nyt $k$-NN luokittimen enemmistösääntö voidaan esittää formaalisti käyttäen luvun 3.1 notaatiota:

$$c_{k\text{NN}}(\mathbf{x}) = \arg \max_{1 \leq j \leq q} (k_j(\mathbf{x})) \tag{3.31}$$

Luokittelua voidaan nopeuttaa esiprosessoimalla opetusjoukko $D$ tietorakenteeksi, josta luokiteltavan vektorin $\mathbf{x} \in \mathbb{R}^d$ lähimmät pisteet löydetään nopeasti. O'Rourke et al. [51] mukaan optimaalisen osajoukon valinta on NP-täydellinen ongelma, joten esiprosessointialgoritmin oikea valinta on tärkeää. Kirjallisuudessa on esitelty erilaisia esiprosessointimenetelmiä hakutietorakenteen muodostamiseksi. Monilla niistä on ei-haluttuja sivuvaikutuksia, kuten suuri muistintarve [1].

Esiprosessointimenetelmäksi tässä työssä valittiin BBD-puu (*balanced box decomposition tree*) menetelmä, jonka ovat esitelleet Arya et al. artikkelissaan *An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions* [1]. BBD-puuta käyttämällä ja hyväksymällä se, että vastauksena saadut naapurit on approksimoitu, saadaan nopeutettua lähimpien naapureiden löytäminen suoritusaikaluokasta $O(kdl)$ suoritusaikaluokkaan $O(kd \log l)$. Menetelmän etuna on myös se, että algoritmin tilatarve pysyy luokassa $O(dl)$. [1]. BBD-puun muodostamistekniikkaa ei esitellä tässä. Aiheesta kiinnostuneet voivat tutustua Aryan et al. [1] algoritmia esittelevään artikkeliin.

Approksimaatiolla tarkoitetaan sitä, kun opetusaineisto $D$ on annettu, ja $\varepsilon > 0$, etsittäessä pisteen $\mathbf{x}$ *todellista* lähintä naapuria $\mathbf{y}^*$, saadaan $(1+\varepsilon)$ -approksimoitu lähin naapuri $\mathbf{y}$.

$$\text{dist}(\mathbf{x}, \mathbf{y}) \leq (1+\varepsilon)\text{dist}(\mathbf{x}, \mathbf{y}^*) \tag{3.32}$$

Toisin sanoen $\varepsilon$ kertoo, mikä on suhteellinen virhe approksimoidun lähimmän naapurin etäisyydessä. Mahdollinen virhe on sama $\varepsilon$ jokaiselle $k$:lle lähinaapurille.

Arya et. al [1] ovat muotoilleet tutkimuksensa tuloksen teoreemaksi 3.

**Teoreema 3** *Otetaan $D_p$, joka on joukko pisteitä $\mathbb{R}^d$ avaruudessa. On olemassa sellainen vakio $c_{d,\varepsilon} \leq d\lceil 1 + 6d/\varepsilon \rceil^d$, että $O(dl \log l)$ ajassa on mahdollista rakentaa $O(dl)$ kokoinen tietorakenne, josta millä tahansa valitulla Minkowski metriikalla voidaan pisteelle $\mathbf{x} \in \mathbb{R}^d$ löytää $k$ $(1+\varepsilon)$ approksimoitua lähintä naapuria suoritus aikaluokassa $O((c_{d,\varepsilon} + kd) \log l)$, kun $\varepsilon > 0$ on annettu ja $1 \leq k \leq l$.*

Approksimoitaessa lähinaapureita virheen mahdollisuus on niin pieni, että käytännön sovelluksissa tällä ei ole merkitystä. Koska BBD-puu menetelmällä on mahdollista laskea myös tarvittaessa oikea lähinaapuri, antaa valittu $k$-NN toteutusmenetelmä liikkumavaraa $\varepsilon$ parametrilla säädettävän hakutarkkuuden muodossa, mikäli lähinaapuriluokittimen suoritusaika osoittautuu ongelmalliseksi.

## 3.4 Sumea luokitin

Vaikka ihmiselle epävarmuuden ja epätäsmällisyyden käsittely on luontevaa, on niiden matemaattinen mallintaminen haastavaa [30, s. 5].

Sumea luokitin on valittu yhdeksi toteutettavista luokittimista juuri ymmärrettävyytensä vuoksi. Isomursu et al. [30, s. 19] korostavat tätä kirjoittamalla, että käytännön sovelluksissa sumeiden systeemien ratkaisevana etuna on ihmisen ajattelua muistuttava lähtökohta. Sumeat päättelysäännöt ja joukot on mahdollista esittää sanallisessa muodossa, jolloin luokittimesta tulee läpinäkyvä ja helppokäyttöinen myös tilastolliseen luokitteluun perehtymättömälle. Toinen käyttökohde sumealle luokittimelle on tapaukset, joissa opetusdataa on saatavilla hyvin vähän tai ei lainkaan.

Sumea luokittelu on sumean teorian (*theory of fuzzy systems*) sovellus. Sumean säädön tapaan sen teoria perustuu sumeisiin joukkoihin (*fuzzy sets*) ja sumeaan logiikkaan (*fuzzy logic*) [55]. Sumea teoria mahdollistaa käsitteiden, kuten epätäsmällisyyden (*vagueness, imprecision, inexactness*) ja epävarmuuden (*uncertainty*) matemaattisen käsittelyn. Työkaluja näiden ilmiöiden mallintamiseen tarvitaan silloin, kun halutaan ongelman mallintamisen sijaan mallintaa asiantuntijan tietämys ratkaistavasta ongelmasta ja kuvata se algoritmina, jolla automatisoidaan ihmisen vastuulla ollut toiminta. [30, s. 4, s. 24]

**Esimerkki.** Sellaiset käsitteet, kuten *kuuma*, *kylmä* ja *sopiva* kuvaavat kukin lämpötilaa. Ilmaisun epätäsmällisyydestä seuraa, että vaikka kahtena peräkkäisenä päivänä sanottaisiin olevan *kylmä*, saattavat päivien mitatut lämpötilat kuitenkin erota toisistaan. Tarkennettaessa käsitteitä saattaisi käydä ilmi, että ensimmäinen päivä oli *todella kylmä* ja toinen *melko kylmä, melkein sopiva*. Vaikka vielä nämäkin käsitteet ovat epätäsmällisiä, ne tarjoavat silti käyttökelpoista informaatiota kyseisen päivän lämpötilasta ja tietoa siitä, miten kyseisinä päivinä olisi kannattanut pukeutua.

Sumeiden systeemien teoria tarjoaa työkaluja epätäsmällisten ilmaisujen matemaattiseen mallintamiseen. Näin esimerkiksi teollisuusprosessin säätöä suorittavalle algoritmille voidaan siirtää prosessin asiantuntijoiden epätäsmällisesti ilmaisemaa tietoa. [30, s. 26]. Sumeiden systeemien teoria on erityisen käyttökelpoinen sellaisissa matemaattisen mallintamisen ongelmissa, joissa pyritään eksaktin ilmiön mallintamisen sijaan mallintamaan ihmisen asiantuntijuutta [30, s. 26]. Sumeat menetelmät ovat myös robusteja. Ihmiset turvaavat usein päätöksiä tehdessään epävarmaan tietoon, jonka avulla voidaan kuitenkin tehdä perusteltuja päätöksiä. Myös epävarma tieto antaa päättelylle tärkeää informaatiota. Jopa itse epävarmuus voi antaa informaatiota varsinaisen mallinnettavan systeemin tilasta. [27]

**Esimerkki.** Systeemin lämpötilasta tiedetään, että tietyllä hetkellä se on 80% todennäköisyydellä 25 °C ± 1 °C. Tätä epävarmuutta voidaan käyttää sumeassa päättelyssä hyödyksi.

Juuri kyky hyväksyä ja hyödyntää häiriöiden vääristämää, epätäsmällistä ja epävarmaa tietoa ongelmaa mallinnettaessa antaa sumeille systeemeille sen tarjoamat edut perinteiseen, konventionaaliseen matemaattiseen mallintamiseen verrattuna [11].

Puolakan mukaan sumea ohjaus voidaan rakentaa operaattorin kokemuksen varaan ilman prosessin matemaattista mallia [55]. Tästä voi laadunvalvontaan liittyvän luokittelun tapauksessa olla paljonkin hyötyä, sillä luokittelukriteerit saattavat olla jossain määrin subjektiivisia, jolloin laaduntarkkailijan luokittelutietämys voidaan siirtää sumeaan järjestelmään ilman ilmiön mallintamista ja luokittelukriteerien eksaktia tuntemista.

Kirjallisuuskatsauksessa löytyi tutkimus, jossa sumeita menetelmiä oli sovellettu konenäkölaadunvalvontaan, joskaan ei luokittelun lähtökohdista. Pandelidis ja Kao [52] kehittivät muovin ruiskuvalun laatuongelmien tietämysjärjestelmän nimeltä Detector, joka perustui sumeisiin relaatioihin. Järjestelmä pyrki auttamaan ruiskuvalukoneen operaattoreita säätämään prosessia oikein järjestelmään tallennetun tietämyksen avulla. Ongelman kuvauksen perusteella Detector ilmoittaa todennäköisimmät syyt vialle ja esittää korjausehdotuksen ongelman poistamiseksi.

Sumeita luokittelumenetelmiä on tutkittu jo 60-luvulta lähtien, minkä vuoksi

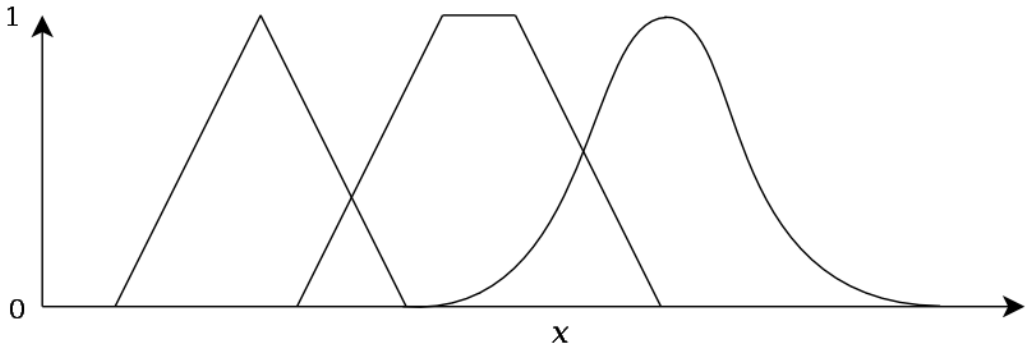
erilaisia sumeita luokittelumenetelmiä on olemassa useita. Osa sumeista luokittelumenetelmistä on toteutettu lisäämällä sumeaa matematiikkaa osaksi aiemmin tunnettuja menetelmiä. Kirjallisuus tuntee useita malleihin perustuvia sumeita luokitusmenetelmiä, sumeita päätöspuu- ja päätösverkkomenetelmiä sekä neurosumeita menetelmiä. Sumeasta hahmontunnistuksesta ja kuvankäsittelystä on olemassa kattava kirja *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*, jossa käsitellään sumeiden menetelmien konenäkösovelluksia paljon laajemmin kuin mihin tässä työssä pystytään. [15] Tässä työssä on valittu sumean luokittimen menetelmäperheistä sääntöpohjaiset IF-THEN -luokittimet ja niiden joukosta esiteltäväksi ja toteutettavaksi Mamdani-tyypin päättelyä käyttävä sumea IF-THEN -luokitin.

Sumeassa logiikassa sallitaan muuttujille muitakin arvoja kuin klassisen logiikan tosi/epätosi. Proposition sumea totuusarvo on suljetulla reaalilukuvälillä $[0, 1]$ [30]. Yleisesti sumea joukko voidaan määritellä siten, että otetaan tavallinen joukko $U$, jota kutsutaan perusjoukoksi. Joukon alkioista $u \in U$ muodostetaan sumea joukko $\tilde{A} \in \mathbb{A}^U$. Merkintä $\mathbb{A}^U$ tarkoittaa kaikkien niiden sumeiden joukkojen joukkoa, jotka voidaan muodostaa perusjoukosta $U$. Perusjoukon $U$ alkioiden $u$ kuulumisen aste (jäsenyysaste, *degree of membership*) joukkoon $\tilde{A}$ ilmaistaan suljetulla reaalilukuvälillä $[0, 1]$ [60]. Jos jäsenyysaste on 1, alkio $u$ kuuluu täysin sumeaan joukkoon $\tilde{A}$, jos se taas on 0, alkio ei kuulu sumeaan joukkoon $\tilde{A}$.

Jäsenyysaste määritellään jäsenyysfunktion $\mu_{\tilde{A}} : U \longrightarrow [0, 1]$ avulla, joka kertoo alkion $u$ jäsenyysasteen sumeassa joukossa $\tilde{A}$. Formaalisti ilmaistuna [55]:

$$\tilde{A} = \{(u, \mu_{\tilde{A}}(u)) \mid \mu_{\tilde{A}}(u) \in [0, 1],\ u \in U\} \tag{3.33}$$

Tyypillisiä sovelluksissa käytettyjä jäsenyysfunktioiden $\mu$ muotoja ovat kolmio, puolisuunnikas ja Gaussin käyrä [55, s. 6] (kuva 3.2).




***Kuva 3.2.*** *Sumeat jäsenyysfunktiot; kolmio, puolisuunnikas ja Gaussin käyrä*

Sumeille joukoille voidaan määritellä joukko-opista tuttuja operaatioita, kuten leikkaus ja yhdiste. Sumean joukko-opin operaatioiden teoriaa esitellään vain niiltä osin kuin se on välttämätöntä yksinkertaisen sumean luokittimen esittelemiseksi.

Sumeiden operaattoreiden määrittelystä kiinnostuneet voivat tutustua niihin Rossin kirjasta *Fuzzy logic with engineering applications* [60].

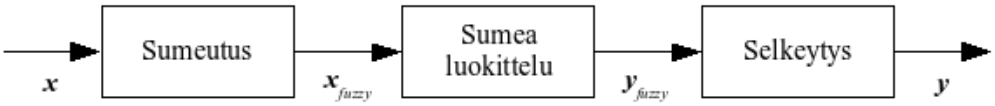
Sumea luokitin käyttää päättelyyn sumeaa lauselogiikkaa. Tässä työssä käytetään Mamdani tyypin IF-THEN säätäjän sumean logiikan operaattorimäärittelyitä, jotka ovat [60]:

$$\mu_{\tilde{A}\wedge\tilde{B}}(x,y) = \min(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(y)) \tag{3.34}$$

$$\mu_{\tilde{A}\vee\tilde{B}}(x,y) = \max(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(y)) \tag{3.35}$$

$$\mu_{\tilde{A}\Rightarrow\tilde{B}}(x,y) = \min(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(y)) \tag{3.36}$$

Sumea luokitin toimii tyypillisesti ympäristössä, jossa sen syöte ja tuloste eivät ole sumeita. Tämän vuoksi varsinainen luokitin tarvitsee avukseen sumeutuksen ja tulkkauksen tekevät osat kuvan 3.3 mukaisesti.




***Kuva 3.3.*** *Sumean luokittelun vaiheet*

Sumeutusta varten tarvitaan vielä sanallisten muuttujien ja termien käsitteet ja valitut sumeiden loogisten operaatioiden määritelmät. Sanallisten muuttujien avulla pyritään mallintamaan arkikielen puheilmaisuja. Se toimii sumeiden joukkojen avulla määriteltävänä vastineena ”selkeän” matematiikan muuttujalle. Niemi [50] määrittelee sanallisen muuttujan seuraavasti:

**Määritelmä 3** *Sanallinen muuttuja on viisikko $(V, T, U, G, M)$, missä $V$ on muuttujan nimi, $T$ on termijoukko, $X$ perusjoukko, $G$ syntaktinen sääntö, jonka mukaan termijoukon $T$ nimet muodostetaan, sekä semanttinen sääntö $M$, joka liittää jokaiseen termiin tietyn merkityksen.*

Termijoukolla $T$ tarkoitetaan muuttujan $V$ sanallisesti määriteltyjä arvoja, kuten ”pieni” ja ”suuri”. Semanttinen sääntö $\tilde{M} : T(V) \longrightarrow \mathbb{A}^U$ puolestaan määrittelee nämä sanalliset termit sumeiden joukkojen avulla. Sanallisella muuttujalla $V$ on siis $|T(V)|$ kappaletta sumeita joukkoja.

$$\forall t \in T(V) \quad \exists\, \tilde{A}_t \in \mathbb{A}^U, \tag{3.37}$$

s.e. $\mu_{\tilde{A}_t}(u))$ kertoo perusjoukon $U$ alkion $u$ ja termin $t$ välisen yhteyden.

Sumeuttamisessa (*fuzzification*) mittausarvot karakterisoidaan sanallisilla muuttujilla laskemalla mittausarvon kuulumisen aste kunkin sanallisen muuttujan termin

sumeaan joukkoon [55]. Syötteen sumeutus perustuu siihen, että kullekin sanalliselle muuttujalle, eli piirrettä vastaavalle sumealle konstruktiolle, voi olla olemassa mikä tahansa määrä termejä. Esimerkiksi pituudelle voi olla määriteltynä termit *lyhyt* ja *pitkä*. Termit määrittävien sumeiden joukkojen avulla propositio "$x_i$ *IS* $\tilde{A}_t$" kuvaa tunnusluvun $x_i$ moniarvologiikan totuusarvoksi välille $[0, 1]$ käyttäen jäsenyysfunktiota $\mu_{\tilde{A}_t}(x_i)$.

Sumeassa päättelyssä muodostetaan propositioista joukko sumeita päättelysääntöjä muotoa "IF $\alpha$, THEN $\beta$". Sumeista päättelysäännöistä koottu joukko muodostaa päättelykoneen sääntökannan. Sääntökannan ei välttämättä tarvitse olla täydellinen. Toisin sanoen kaikissa säännöissä ei tarvitse esiintyä kaikkia muuttujia eikä muuttujilla tarvitse olla atomisääntöä kaikille sen sumeille joukoille. Mikäli kaikki säännöt kuitenkin on määritelty, sanotaan sääntökannan olevan tällöin täysi. Etujäsen $\alpha$ koostuu luokittimen tapauksessa propositioista muotoa "$x_1$ *kuuluu sumeaan joukkoon* $\tilde{A}_1$". Koko etujäsen on siis muotoa "$x_1$ *kuuluu sumeaan joukkoon* $\tilde{A}_1$ AND $x_2$ *kuuluu sumeaan joukkoon* $\tilde{A}_2$ ... AND $x_d$ *kuuluu sumeaan joukkoon* $\tilde{A}_d$".

Säännöstö pitää sisällään luokitteluun tarvittavan tietämyksen. Säännöt evaluoidaan sijoittamalla sumeutetut mittausarvot, jonka jälkeen lopullinen luokitus lasketaan selkiytysalgoritmin avulla. [55].

**Esimerkki.** Luokittimelle syötteen tuottaviksi piirreilmaisimiksi on valittu löydetyn virheen *pituutta* ja *pitkänomaisuutta* (leveyden ja pituuden suhde) mittaavat piirreilmaisimet. Operaattori määrittää pituudelle sopivasti valittujen puolisuunnikkaan muotoisten jäsenyysfunktioiden avulla sanallisille arvoille sumeat joukot *pitkä* ja *lyhyt*. Pitkänomaisuudelle hän määrittää sanallisille arvoille *pyöreähkö*, *ovaalimainen*, *viivamainen* niitä vastaavat sumeat joukot.
Tämän jälkeen operaattori määrittää säännöstön seuraavasti:

1. *"Jos pituus on pitkä ja pitkänomaisuus on viivamainen, niin luokka on naarmu"*
2. *"Jos pituus on lyhyt ja pitkänomaisuus on pyöreähkö, niin luokka on tahra"*
3. *"Jos pituus on lyhyt ja pitkänomaisuus on ovaalimainen, niin luokka on tahra"*

Tämän säännöstön säännöt evaluoimalla ja maksimi jäsenyysasteisen tuloksen valitsemalla luokitin osaa määrittää virheelle luokan.

Sumean luokittimen toimintaa ohjaavat ehdot talletetaan sääntökantaan. Sääntökannassa voi olla $N \in \{(q \dots q \prod |A|\})$ sääntöä, missä $|A|$ on sumean luokittimen tietämyksen mallintamiseen käytettyjen sumeiden joukkojen lukumäärä. Sääntöjen määrää on rajattu, sillä $q$:n luokan luokittelutehtävässä kunkin luokituksen tulee olla mahdollinen ja toisaalta sääntöjen enimmäismäärän rajaa mahdollisten propo-

sitioyhdistelmien lukumäärä. Sääntökannan muodostavat sumeat IF-THEN säännöt $R_k \quad k = 1 \ldots N$ voidaan kirjoittaa muodossa

$$\text{Rule } R_k : \text{IF } x_1 \text{ IS } \tilde{A}_{k1} \text{ AND } \ldots \text{ AND } x_d \text{ IS } \tilde{A}_{kd} \text{ THEN } y \text{ IS } \tilde{C}_k, \tag{3.38}$$

missä $N$ on sääntökannan sääntöjen määrä. Sisääntulosuureiden sumeuttamiseen käytetyt sumeat joukot $\tilde{A}_{ki}$ ovat $i$:nnen sanallisen muuttujan $k$:nnessa säännössä käytetyn termin edustajia. Lähtösuureen sumeana joukkona $\tilde{C}_k$ käytetään tässä työssä yksiöfunktioita $\mu_{\tilde{C}_k}$, jotka voivat edustaa jotain luokittelun kohdeluokkaa $\omega_j, j = 1 \ldots q$. Tässä työssä esitelty sumea luokitintyyppi käyttää siis luokitusta säännön seurauksena [11].

$$\mu_{\tilde{C}_j}(y) = \begin{cases} 1, & y = j \quad (3.39) \\ 0, & \text{muuten} \quad (3.40) \end{cases}$$

Yhtä luokkaa $\omega_j$ ja sen luokkatunnistetta $j$ kohden voi olla useita sääntöjä. Jokainen sääntö voi kuitenkin implikoida kuuluvuutta vain yhteen luokkaan. Luokiteltaessa kohdetta, jonka piirrevektori on $\mathbf{x}$, evaluoidaan kunkin säännöstön säännön liipaisuvoimakkuus (*firing strength*) kullekin luokitukselle $y$. Käyttämällä sumeiden loogisten operaattoreiden määrittelyjä (3.34)-(3.36), säännön $R_k$ evaluointi voidaan kirjoittaa auki seuraavasti:

$$\mu_{\tilde{R}_k}(\mathbf{x}, y) = \min\left\{\mu_{\tilde{C}_k}(y), \min\left\{\mu_{\tilde{A}_{k1}}(x_1), \mu_{\tilde{A}_{k2}}(x_2), \ldots, \mu_{\tilde{A}_{kd}}(x_d)\right\}\right\} \tag{3.41}$$

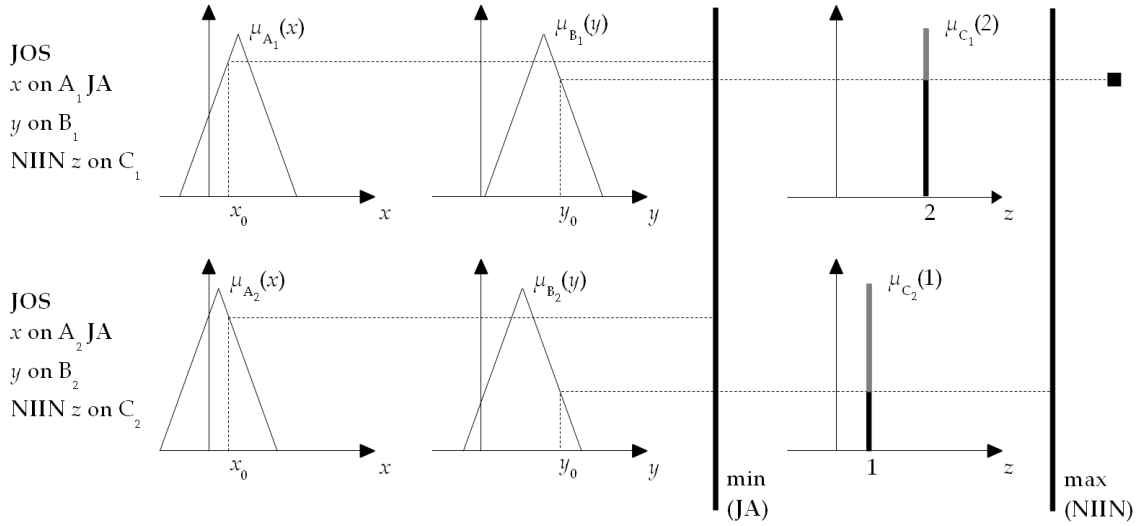
Sisempi min-funktio toteuttaa säännön AND-yhdistelyn. Koska lähtösuureena toimivan luokituksen sumeina joukkoina käytetään tässä työssä yksiöfunktiota (*singleton function*), niin ulompi min-funktio valikoi ne säännöt, jotka antavat tulokseksi luokituksen $y$.

Säännöt yhdistetään Mamdanin sumeassa päättelyssä käyttäen min-max-kompositiota [55, s. 36-38]. Kompositiosta seuraa, että luokkakohtaiset sumean päättelyn päätösfunktiot voidaan ilmaista muodossa:

$$g_y(\mathbf{x}) = \max\left\{\mu_{\tilde{R}_1}(\mathbf{x}, y), \mu_{\tilde{R}_2}(\mathbf{x}, y), \ldots, \mu_{\tilde{R}_N}(\mathbf{x}, y)\right\} \tag{3.42}$$

Päätösfunktio $g_y(\mathbf{x})$ kertoo näytteen $\mathbf{x}$ kuulumisen asteen luokkaan, jonka tunniste on $y$. Nyt sumea luokitin $c_{\text{fuzzy}}$ voidaan luvun 3.1 luokitinnotaatiota käyttäen esittää muodossa:

$$c_{\text{fuzzy}}(\mathbf{x}) = \arg\max_{1 \le i \le q}(g_i(\mathbf{x})) \tag{3.43}$$

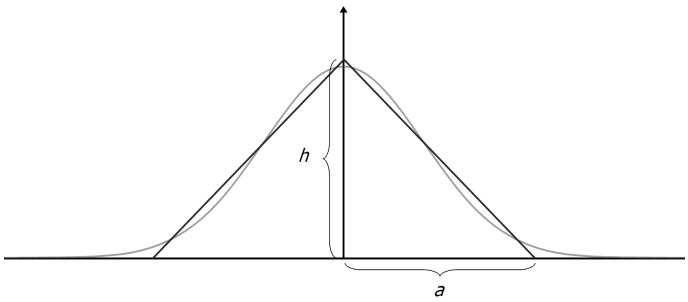



***Kuva 3.4.*** *Kahden säännön ja kahden luokan sumea Mamdani-luokittelutehtävä.*

Edellä esiteltyä Mamdanin päättelyyn perustuvaa luokitusmenetelmää on pyritty havainnollistamaan kuvassa 3.4. Kuvassa kahden luokan, kahden piirteen ja kahden säännön sumea Mamdani-luokitin laskee maksimaalisen jäsenyysasteen sääntökannan säännöille. Ensimmäisessä vaiheessa AND-operaattoreiden yhdistämät propositiot yhdistetään min-operaatiolla, toisessa vaiheessa luokituskohtaiset johtopäätöksen jäsenyysasteet yhdistetään premissin jäsenyysasteen kanssa ja lopuksi valitaan sen säännön jäsenyysaste, joka antaa korkeimman tuloksen max-operaatiolla.

Sumean luokittimen opettamiseen on sovellettu lukuisia eri menetelmiä, joita Roubos et al. [61] listaavat artikkelissaan *Learning fuzzy classification rules from labeled data*. Suosittuja ovat olleet muun muassa neurosumeat menetelmät, klusterointimenetelmät ja geneettiset algoritmit. Menetelmäkatsauksen lisäksi Roubos et al. esittelevät klusterointia, separoituvuusanalyysiä, sääntöjen similaarisuutta ja geneettisiä algoritmeja yhdistelevän sumeiden IF-THEN -luokittimien opetusmenetelmän, joka tuottaa tiiviin sääntökannan ja pienen sumeiden joukkojen lukumäärän. Roubos et al. kuvaamaa opetusmenetelmä ei tässä työssä toteutettu, joten sitä ei esitellä sen tarkemmin. Tässä työssä sumean luokittimen opettamiseen käytetään yksinkertaistettua menetelmää, jossa luvussa 3.2 esiteltyä Bayes-luokittimen normaalijakaumaestimaattoria käytetään sumeiden joukkojen määrittämiseen.

Lähtökohdaksi otetaan oletuksen (3.11) mukaiset normaalijakautuneet luokka- ja piirrekohtaiset tiheysfunktioestimaatit. Kuten luvussa 3.2 on kuvattu, estimaatit saadaan laskemalla luokka- ja piirrekohtaiset keskiarvot $\bar{x}_{ij}$ ja keskihajonnat $\sigma_{ij}$, missä $i \in 1 \ldots d$ on piirreindeksi ja $j \in 1 \ldots q$ luokkatunniste. Huomaa, että tässä osiossa opetusaineistosta laskettuja keskiarvoja merkitään $\bar{x}$:lla, sillä merkintää $\mu$ käytetään sumeista jäsenyysfunktioista.




***Kuva 3.5.*** *Normaalijakauman tiheysfunktion approksimointi kolmiofunktion avulla.*

Luokka- ja piirrekohtaisia normaalijakaumia voidaan approksimoida kolmiofunktioilla $f_{\mathrm{T}}$. Approksimaatiota varten meidän tulee laskea kolmiofunktion korkeus $h$ ja kannan puolikas $a$ (kuva 3.5). Määritellään kolmiofunktio joka saa suurimman arvonsa kun $x = 0$ yhtälöryhmällä:

$$f_{\mathrm{T}}(x) = \begin{cases} x\dfrac{h}{a} + h, & x \in [-a, 0] \quad (3.44) \\ -x\dfrac{h}{a} + h, & x \in (0, a] \quad (3.45) \\ 0, & x \notin [-a, a] \quad (3.46) \end{cases}$$

Kolmiofunktion parametrit $h$ ja $a$ saadaan yhtälöryhmästä, jossa kolmiofunktion ja normaalijakauman tiheysfunktion (3.12) pinta-alat ja toiset keskusmomentit on kiinnitetty yhtäsuuriksi. Jakaumien keskiarvo on 0, joten ne ovat pystyakselin suhteen symmetrisiä.

$$\begin{cases} \displaystyle\int_{-\infty}^{\infty} f_{\mathrm{T}}(x)\,dx = \int_{-\infty}^{\infty} f_{\mathrm{NDF}}(x)\,dx & (3.47) \\ \displaystyle\int_{-\infty}^{\infty} x^2 f_{\mathrm{T}}(x)\,dx = \int_{-\infty}^{\infty} x^2 f_{\mathrm{NDF}}(x)\,dx & (3.48) \end{cases}$$

$$\begin{cases} \displaystyle 2\int_0^a (-x\frac{h}{a} + h) = \int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-x^2/2\sigma^2}\,dx & (3.49) \\ \displaystyle 2\int_0^a x^2(-x\frac{h}{a} + h)\,dx = \int_{-\infty}^{\infty} x^2 \frac{1}{\sigma\sqrt{2\pi}} e^{-x^2/2\sigma^2}\,dx & (3.50) \end{cases}$$

Normaalijakauman tiheysfunktion määrätty integraali antaa määritelmänsä mukaisesti pinta-alaksi 1. Myöskään normaalijakauman tiheysfunktion momentteja ei johdeta tässä, vaan ne oletetaan tunnetuiksi. Integroimalla yhtälöryhmän toisen yhtälön vasemman puolen symmetrisen polynomifunktion ja korvaamalla normaalijakauman tiheysfunktion toisen keskusmomentin kaava sen arvolla $\sigma^2$ [37] yhtälöryhmä saadaan muotoon:

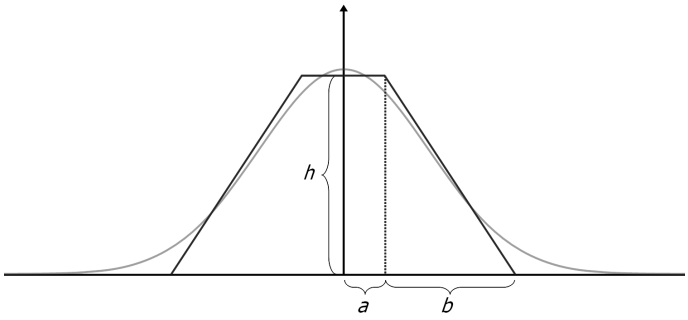
$$\begin{cases} ha = 1 & (3.51) \\ \dfrac{a^3 h}{6} = \sigma^2, & (3.52) \end{cases}$$

josta voidaan ratkaista kolmiofunktion korkeus $h$ ja kannan puolikas $a$:

$$\begin{cases} a = \sqrt{6}\sigma & (3.53) \\ h = \dfrac{1}{a} & (3.54) \end{cases}$$

Kun jakauman keskiarvo $\bar{x}$ ja keskihajonta $\sigma$ tunnetaan, voidaan normaalijakauman tiheysfunktion kolmiofunktioapproksimaatiosta johdettu jäsenyysfunktio esittää muodossa:

$$\hat{\mu}_{\tilde{F}}(x) = \begin{cases} 0, & x < \bar{x} - \sqrt{6}\sigma \quad (3.55) \\ \dfrac{(x-\bar{x}) + \sqrt{6}\sigma}{6\sigma^2}, & \bar{x} - \sqrt{6}\sigma \leq x < \bar{x} \quad (3.56) \\ \dfrac{-(x-\bar{x}) + \sqrt{6}\sigma}{6\sigma^2}, & \bar{x} \leq x < \bar{x} + \sqrt{6}\sigma \quad (3.57) \\ 0, & x \geq \bar{x} + \sqrt{6}\sigma \quad (3.58) \end{cases}$$




***Kuva 3.6.*** *Normaalijakauman tiheysfunktion approksimointi puolisuunnikasfunktion avulla.*

Toinen sumeassa säädössä yleisesti käytetty jäsenyysfunktion muoto on puolisuunnikasfunktio. Normaalijakauman puolisuunnikasapproksimaatio voidaan johtaa samalla tavalla kuin se tehtiin kolmiofunktiolle. Approksimaatiolle tulee määrittää puolisuunnikasfunktion korkeus $h$, funktion laen leveyden puolikas $a$ ja kyljen kanta $b$ (kuva 3.6. Korkeus saadaan esitettyä parametrien $a$ ja $b$ avulla kun merkitään normaalijakauman tiheysfunktion puolisuunnikasapproksimaation pinta-alaksi 1.

$$2ha + hb = 1 \;\Leftrightarrow\; h = \frac{1}{2a + b} \quad (3.59)$$

Määritellään symmetrinen puolisuunnikasfunktio seuraavasti:

$$f_{\mathrm{Z}}(x) = \begin{cases} \dfrac{x+a+b}{2ab+b^2}, & x \in [-a-b, -a] \quad (3.60) \\ \dfrac{1}{2a+b}, & x \in (-a, a] \quad (3.61) \\ \dfrac{-x+a+b}{2ab+b^2}, & x \in (a, a+b] \quad (3.62) \\ 0, & x \notin [-a-b, a+b] \quad (3.63) \end{cases}$$

Muodostetaan yhtälöryhmä käyttäen puolisuunnikasfunktion ja normaalijakauman tiheysfunktion toisia ja neljänsiä keskusmomentteja.

$$\begin{cases} \displaystyle\int_{-\infty}^{\infty} x^2 f_{\mathrm{Z}}(x)\,dx = \int_{-\infty}^{\infty} x^2 f_{\mathrm{NDF}}(x)\,dx & (3.64) \\ \displaystyle\int_{-\infty}^{\infty} x^4 f_{\mathrm{Z}}(x)\,dx = \int_{-\infty}^{\infty} x^4 f_{\mathrm{NDF}}(x)\,dx & (3.65) \end{cases}$$

Samoin kun kolmiofunktion tapauksessa, voimme käyttää jakaumien symmetriaa hyväksi:

$$\begin{cases} 2(\displaystyle\int_0^a x^2 \frac{1}{2a+b}\,dx + \int_a^{a+b} x^2 \frac{-x+a+b}{2ab+b^2}\,dx) = \int_{-\infty}^{\infty} x^2 \frac{1}{\sigma\sqrt{2\pi}} e^{-x^2/2\sigma^2}\,dx & (3.66) \\ 2(\displaystyle\int_0^a x^4 \frac{1}{2a+b}\,dx + \int_a^{a+b} x^4 \frac{-x+a+b}{2ab+b^2}\,dx) = \int_{-\infty}^{\infty} x^4 \frac{1}{\sigma\sqrt{2\pi}} e^{-x^2/2\sigma^2}\,dx & (3.67) \end{cases}$$

Yhtälöryhmän yhtälöiden vasemmat puolet saadaan laskemalla polynomifunktioiden määrätyt integraalit. Oikeat puolet puolestaan voidaan korvata tunnetuilla tiheysfunktion ominaisuuksilla [37].

$$\begin{cases} \dfrac{1}{6}(2a^2+2ab+b^2) = \sigma^2 & (3.68) \\ \dfrac{1}{15}(3a^4+6a^3b+7a^2b^2+4ab^3+b^4) = 3\sigma^4 & (3.69) \end{cases}$$

Tälle yhtälöryhmälle ei löydy reaalista ratkaisua, joten parametrien $a$, $b$ ja $h$ suhteille ei ole analyyttistä ratkaisua. Tässä työssä päädyttiin merkitsemään $b = a$, jolloin emme tarvitse neljänsiä keskusmomentteja, vaan voimme ratkaista parametrin $a$ toisten keskusmomenttien avulla.

$$\frac{1}{6}(2a^2 + 2ab + b^2) = \sigma^2 \quad \| b = a \tag{3.70}$$

$$\frac{5a^2}{6} = \sigma^2 \tag{3.71}$$

$$a = \sqrt{\frac{5}{6}}\sigma \tag{3.72}$$

Jos yllä esitellyt jakaumien approksimaatiot halutaan muuttaa sumeiksi jäsenyysfunktioiksi, jotka saavat kaikki arvot väliltä $[0, 1]$, voidaan ne normalisoida. Skaalaus heikentää luokittelutarkkuutta, mutta tekee sumeista joukoista standardimuotoisen sumean säätäjän sumeita joukkoja. Skaalaus tapahtuu jakamalla approksimaatiofunktio $\hat{\mu}_{\tilde{F}_{ij}}$ sen huippukohdalla.

$$\mu_{\tilde{A}_{ij}}(u) = \frac{\hat{\mu}_{\tilde{F}_{ij}}}{\max_{u \in U} \hat{\mu}_{\tilde{F}_{ij}}(u)}. \tag{3.73}$$

Kun sumeat joukot on määritelty, voidaan kullekin luokitukselle laatia yksi päättelysääntö. Piirre- ja luokkakohtaisista jakaumaestimaateista (piirreindeksille $i \in 1 \ldots d$ ja luokkaindeksille $j \in 1 \ldots q$) johdetut sumeat joukot $\tilde{A}_{ji}$ toimivat piirrettä vastaavan sanallisen muuttujan $V_i$ termien semanttiset säännöt $M_i$ määrittävinä sumeina joukkoina. Ne voidaan järjestää keskiarvon $\mu$ mukaiseen järjestykseen, jonka jälkeen sanallisen muuttujan syntaktinen sääntö $G_i$ voi generoida termeille $T(V_i)$ nimet. Tämän yksinkertaistetun menetelmän generoima sääntökanta voidaan ilmaista aiemmin esiteltyjen notaatioiden ja tiheysfunktioista johdettujen sumeiden joukkojen avulla seuraavasti:

$$\text{Rule } R_j : \text{IF } x_1 \text{ IS } \tilde{A}_{j1} \text{ AND } \ldots \text{ AND } x_d \text{ IS } \tilde{A}_{jd} \text{ THEN } y \text{ IS } j, \quad \forall j \in Y \tag{3.74}$$

Jakauma-approksimaatioiden avulla rakennettua sääntökantaa voidaan käyttää lähtökohtana rakennettaessa virheluokittelijaa asiantuntijan ohjeiden mukaisesti. Sääntöjä voidaan lisätä tai poistaa, ja sumeiden joukkojen jäsenyysfunktioiden paikkaa ja muotoa voidaan muuttaa. Yllä kuvattu sumean luokittimen opetusmenetelmä ei ole riittävän hyvä, jotta se mahdollistaisi sumeiden luokittimien hyvän luokittelutarkkuuden pelkästään opetusaineistosta johdetuilla sumeilla joukoilla ja sääntökannalla.

## 3.5 SVM-luokitin

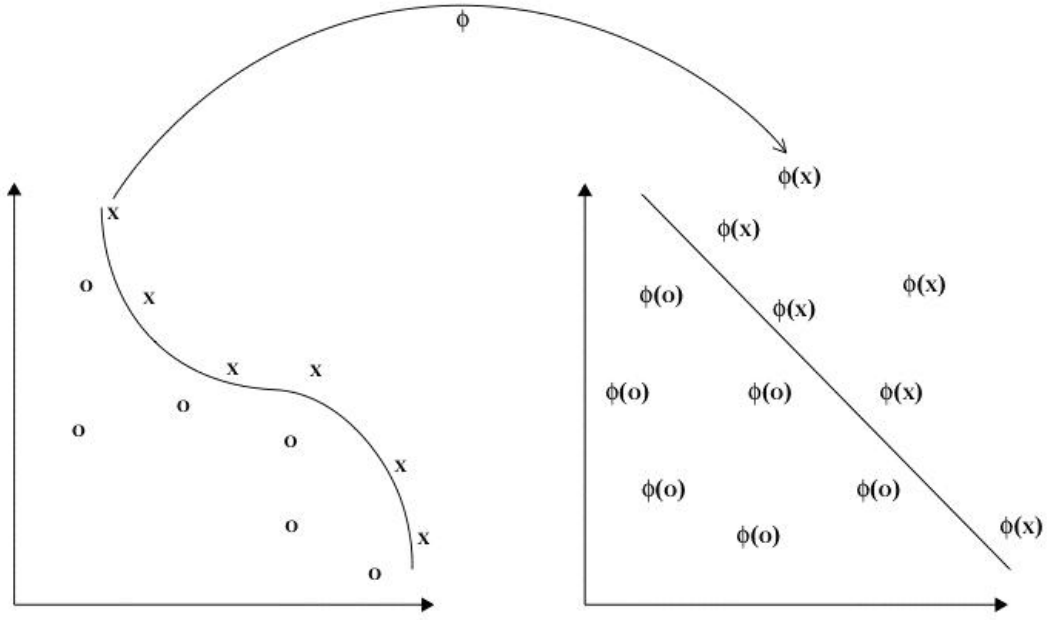
SVM- eli tukivektorikonemenetelmät (*Support Vector Machines*) ovat vuonna 1992 tapahtuneen esittelynsä [2] jälkeen olleet jatkuvan tutkimuksen kohteena. SVM-luokittimia on myös menestyksekkäästi sovellettu monissa käytännön sovelluksissa [56; 32].

Tukivektorikone jakaa piirreavaruuden hypertasoilla osiin niin, että eri luokkiin kuuluvat näytteet ovat eri puolilla hypertasoja. Hypertasot saadaan käyttäen lineaarisen optimoinnin menetelmiä siten, että luokitteluvirheet minimoidaan ja luokkien väliin jäävät marginaalit maksimoidaan. Hypertasot muodostavat piirreavaruuteen osajoukkoja, jotka edustavat kohdeluokkia. Hypertason sovittamisessa opetusjoukon pisteiden avulla voidaan käyttää joko optimimarginaaleja tai joustavia marginaaleja. [13]

SVM-luokittimien menestyksellisyys piilee niin sanotun kernelitempun käytössä (*kernel trick*). Kernelitemppu vastaa lähtödatan keinotekoista kuvausta (yleensä) korkeampiulotteiseen avaruuteen. Piirreavaruuden vektoreiden kuvausta toiseen esitysmuotoon merkitään funktiolla:

$$\Phi : \mathbb{R}^d \longrightarrow \mathbb{R}^N. \tag{3.75}$$

Kuvauksen vaikutusta luokitteluongelmaan havainnollistetaan kuvassa 3.7. Kuvassa on kuvattu opetusdatapisteet kohdeavaruuteen.




***Kuva 3.7.*** *Opetusdatan kuvaus piirreavaruudesta kohdeavaruuteen, jolloin epälineaarisesta luokittelutehtävästä on saatu lineaarinen. [13]*

Kernelifunktio määritellään kuvauksen $\Phi$ ja vektoreiden sisätulon avulla seuraavasti:

$$K(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}) \rangle \tag{3.76}$$

Luokitteluongelmissa, missä piirreavaruuden dimensio on suuri saattaa riittää lineaarikuvaus korkeampidimensionaaliseen avaruuteen. Tällöin voidaan käyttää lineaarista kerneliä:

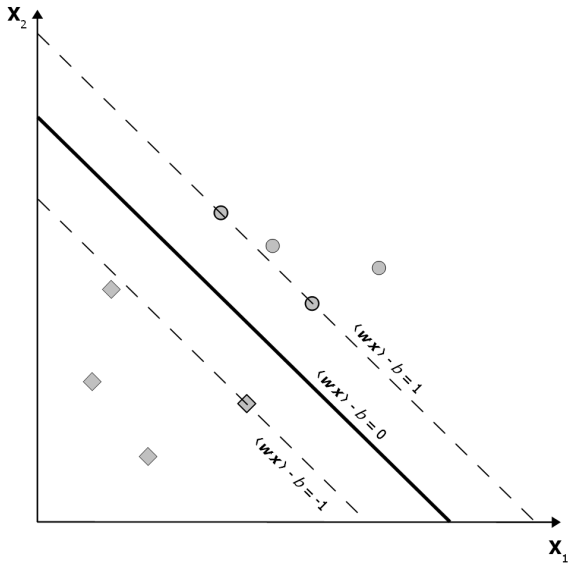
$$K_{\text{linear}}(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle. \tag{3.77}$$

Epälineaarista kuvausta tarvitseville luokittelutehtäville hyvä kernelivalinta on RBF-kerneli (*Radial Basis Function*) [25]

$$K_{\text{RBF}}(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2), \tag{3.78}$$

missä $\gamma > 0$ on kernelin parametri.

Sopivasti valitun kernelin ja sillä suoritetun kernelitempun avulla epälineaarisia hypertasoja vaativat luokitteluongelmat voidaan ratkaista lineaarisen luokittelun menetelmin [2; 13]. Lineaarisen luokittimen opetuksen tuloksena kernelin avulla tehdyn kuvauksen kohdeavaruus on jaettu osiin muotoa $(w) \cdot (x) - b = 0$ olevilla hypertasoilla (kuva 3.8).




***Kuva 3.8.*** *Hypertasolla määritelty maksimimarginaaliluokitin*

Oletetaan ensin kahden luokan tapaus ja sille opetusjoukko $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$, jossa opetusvektorit $\mathbf{x}_i \in \mathbb{R}^d, i = 1 \ldots l$ ja luokkatunnisteet $y_i \in \{1, -1\}$. Luokkatunnisteet voidaan merkitä myös vektorimuodossa $\mathbf{y} \in \{1, -1\}^l$. Opetusjoukon piirrevektoreita $\mathbf{x}_i$, joille pätee $y_i(\mathbf{x} \cdot \mathbf{x}_i + b) = 1$ kutsutaan tukivektoreiksi.

Boser et al. [2] esittelemä ja Cortesin ja Vapnikin [12] pehmeillä marginaaleilla laajentama kahden luokan $C$-tukivektorikoneluokittimien ($C$-SVC) luokitteluongelma voidaan formuloida lineaarisen optimoinnin ongelmaksi [8].

Ongelman primaalimuoto [12]:

$$\begin{array}{rl} \min_{\mathbf{w},b,\xi} & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{ehdoilla} & y_i(\mathbf{w}^T\Phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \qquad i = 1 \ldots l. \end{array} \tag{3.79}$$

Pelivaramuuttujavektori $\xi$ (*slack-variables*) varmistaa, että myös ei-separoituvan datan päätöspinnan sovitustehtävälle löytyy ratkaisu. Käytännössä primaalimuotoa käyttökelpoisempi on vastaava duaalitehtävä [8]:

$$\begin{array}{rl} \min_{\alpha} & \frac{1}{2}\alpha^T Q\alpha - \mathbf{e}^T\alpha \\ \text{ehdoilla} & \mathbf{y}^T\alpha = 0, \\ & 0 \leq \alpha_i \leq C, \qquad i = 1 \ldots l, \end{array} \tag{3.80}$$

missä $\mathbf{e}$ on vektori, jonka kaikki arvot ovat ykkösiä. Pelivaramuuttujien merkityksellisyyttä säätelevä $C > 0$ toimii duaalitehtävässä minimoitavan $\alpha$-muuttujan ylärajana. $Q$ on $l \times l$ positiivinen semidefiniitti matriisi, jonka alkiot saadaan kernelifunktion $K(\mathbf{x}_i, \mathbf{x}_j)$ avulla laskemalla $Q_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$.

Piirrevektorit korkeampiulotteiseen avaruuteen kuvaavan $\Phi$ ansiosta luokittelu voidaan suorittaa alla olevan lineaarisen päätösfunktion avulla.

$$c(\mathbf{x}) = sgn\left(\sum_{i=1}^{l} y_i\alpha_i K(\mathbf{x}_i, \mathbf{x}) + b\right) \tag{3.81}$$

Schölkopf et al. [62] esittelivät vuonna 2000 uudentyyppisen tukivektorikoneluokittimen, joka tunnetaan nimellä $\nu$-SVC. Parametri $\nu \in (0, 1]$ ohjaa opetusvirheen ylärajaa ja määrittää tukivektoreiden määrän [8]. $\nu$-SVC luokittimen etuna Cortesin ja Vapnikin [12] esittelemään $C$-SVC luokittimeen on parametrin $\nu$ sopivan arvon löytämisen helppous verrattuna parametrin $C$ oikean arvon löytämiseen [7]. $\nu$-SVC luokittimen primaalimuoto on [7]:

$$\begin{array}{rl} \min_{\mathbf{w},b,\xi,\rho} & \frac{1}{2}\mathbf{w}^T\mathbf{w} - \nu\rho + \frac{1}{l}\sum_{i=1}^{l}\xi_i \\ \text{ehdoilla} & y_i(\mathbf{w}^T\Phi(\mathbf{x}_i) + b) \geq \rho - \xi_i, \\ & \rho \geq 0, \xi_i \geq 0, \qquad i = 1 \ldots l. \end{array} \tag{3.82}$$

Ja sen duaali [7]:

$$\begin{array}{rl} \min_{\alpha} & \frac{1}{2}\alpha^T Q\alpha \\ \text{ehdoilla} & 0 \leq \alpha_i \leq 1/l, \qquad i = 1 \ldots l, \\ & \mathbf{e}^T\alpha \geq \nu, \quad \mathbf{y}^T\alpha = 0. \end{array} \tag{3.83}$$

$\nu$-SVM luokittimen matriisi $Q$ ja päätösfunktio $c(\mathbf{x})$ määritellään kuten $C$-SVC:n tapauksessa.

LP-duaaliongelman ratkaisuna saatavan $\alpha$ vektorin päätöspinnan määrittävä vektorin $\mathbf{w}$ välillä on yhteys, joka voidaan ilmaista kuvausfunktiota $\Phi$ käyttäen seuraavasti:

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \Phi(\mathbf{x}_i) \tag{3.84}$$

Luokiteltaessa vektoria $\mathbf{x}$ sisätulot voidaan laskea kernelitempun avulla seuraavasti:

$$\langle \mathbf{w} \cdot \Phi(\mathbf{x}) \rangle = \sum_{i=1}^{l} \alpha_i c_i K(\mathbf{x}_i, \mathbf{x}) \tag{3.85}$$

Yllä esitellyt LP formuloinnit johtavat quadraattisiin optimointiongelmiin, joiden ratkaisemiseksi käytetään SMO-tyyppisiä dekompositiomenetelmiä (*Sequental Minimal Optimization*) yhdessä numeeristen kuristus- (*shrinking*) ja välitulosmenetelmien (*caching*) kanssa. SMO-algoritmiin ja sen apuna käytettyihin ratkaisumenetelmiin voi tutustua lähteissä Chang ja Lin 2009 [8] ja Cristianini 2000 [13, s. 137].

Kun luokkia on useampia kuin kaksi, Chang ja Lin [8] ehdottavat käytettäväksi Knerr et al. [35] esittelemää ”yksi-vastaan-yksi” -menetelmää. Kun luokkia on $q$ kappaletta, menetelmässä rakennetaan $q(q-1)/2$ luokitinta, joista kukin toteuttaa kahden luokan välisen päätöspinnan.

Tässä luokkatunnisteita käytetään kuten aiemmin, mutta yleistettynä $q$:n luokan tapaukseen merkitsemällä $y_i \in 1 \dots q$. Kunkin luokittimen rakentamiseksi kootaan opetusjoukko $D_{ab} = \{(\mathbf{x}, y_{ab}(y)) : (\mathbf{x}, y) \in D, (y = a \vee y = b)\}$, missä

$$y_{ab}(y) = \begin{cases} 1 : y = a \\ -1 : y = b \end{cases}$$

Muotoa $D_{ab} \subset \mathbb{R}^d \times [-1, 1]$ olevilla opetusjoukoilla voidaan nyt opettaa kaikki mahdolliset luokkaparit $a \neq b$ käyttäen LP ongelmaa (3.79) tai (3.82).

Luokiteltaessa piirrevektoria $\mathbf{x}$ luokittelupäätös lasketaan kaikille luokittimelle kaavan (3.81) avulla. Luokittelupäätös antaa äänen joko luokkatunnisteen $a$ tai $b$ edustamalle luokalle. Se luokka, jolla on eniten ääniä, on luokittelun tulos. Mikäli kaksi tai useampi luokka saa saman verran ääniä, valitaan se, jonka tunniste on pienin. Vaihtoehtoisista usean luokan SVM-luokittelumenetelmistä kiinnostuneen kannattaa tutustua Hsun ja Linin tekemään menetelmävertailuun [24].

## 3.6 Luokittimien vertailumenetelmät

Luokittimen luokittelutarkkuutta voidaan mitata luokituksen virhetodennäköisyyden avulla. Virheluokitustodennäköisyyden analyyttinen johtaminen luokittimen matemaattisista perusteista on hyvin vaikeaa, joten tässä työssä sitä arvioitiin tilastollisin menetelmin [70]. Tässä työssä virheluokitustodennäköisyksien arviointiin käytetään standardia ristiinvalidointitestiä (*standard cross validation*, SCV) [70, s. 254]. Menetelmässä luokittimen opetusjoukko $D_i$ saadaan kun poistetaan valitusta opetusjoukosta $D$ yksi näyte $(\mathbf{x}_i, y_i)$.

$$D_i = D \setminus \{(\mathbf{x}_i, y_i)\} \tag{3.86}$$

Kun luokitin on opetettu opetusjoukolla $D_i$, luokitellaan näyte $\mathbf{x}_i$ ja tarkistetaan vastaako luokitus tunnistetta $y$. Poistamalla ja luokittelemalla kukin opetusjoukon näyte vuorollaan, voidaan laskea virheluokitusprosentti. Formaalisti standardi ristiinvalidointitesti voidaan esittää seuraavasti:

**Teoreema 4** *Kun $N = |D|$ ja apufunktio $Q$ on*

$$Q(y, c(\mathbf{x}; D_i)) = \begin{cases} 0, & josy = c(x; D_i) \quad (3.87) \\ 1, & muuten, \quad (3.88) \end{cases}$$

*voidaan standardin ristiinvalidointitestin antama virhetodennäköisyyden estimaatti ilmaista muodossa:*

$$e_{SCV} = \frac{1}{N} \sum_{j=1}^{N} Q(y_j, c(\mathbf{x}_j; D_j)) \tag{3.89}$$

Ristiinvalidointi voidaan myös tehdä jakamalla aineistojoukko $D$ $v$:hen yhtäsuureen osajoukkoon. Kukin osajoukoista $T_i,\ i \in 1 \ldots v$ toimii vuorollaan testijoukkona, joka luokitellaan opetusjoukolla $D_i = D \setminus T_i$ opetetulla luokittimella. Tällöin puhutaan $v$-kertaisesta ristiinvalidoinnista (*$v$-fold cross validation*, VCV). [70, s. 255]

Vaihtoehtoisia menetelmiä ristiinvalidoinnille virheluokitustodennäköisyyden estimoimiseksi olisivat olleet takaisinsijoitus- (*resubstitution*), pidätys- (*holdout*), jackknife- ja bootstrap-menetelmät. Työssä päädyttiin kuitenkin ristiinvalidointiin, sillä se antaa suhteellisen harhattoman estimaatin luokitteluvirhetodennäköisyydelle. Toinen ristiinvalidointimenelmän etu on, että se soveltuu hyvin myös luokittimien parametrien valinnan ja säätämisen automatisointiin.

# 4. TUTKIMUSMENETELMÄT JA TOTEUTUS

Aliluvussa 4.1 kuvataan työn aikana toteutetun luokittelijaohjelmiston rakenne ja sen kehittämisessä käytetyt menetelmät ja ohjelmistokirjastot. Aliluvussa 4.2 esitellään tässä työssä luokittelijoiden testaamiseen käytetyt testiaineistot. Luvun viimeisessä aliluvussa määritellään käytetyt testausmenetelmät ja perustelut sille, miksi kyseiset testit ovat olennaisia tässä työssä.

## 4.1 Luokittelijaohjelmiston toteutus

Ohjelmiston kehittäminen työn kuluessa oli tilaajan kannalta tärkeää. Työn ohjelmistokehitysvaiheeseen kuului luokitteluohjelmiston käyttöliittymän rakentaminen, luokittimien toteutus ja integrointi sekä testiohjelmien suunnittelu ja toteuttaminen. Ohjelmiston yleiskäyttöisyyteen ja integroitavuuteen kiinnitettiin erityistä huomiota. Toteutusvaiheen aikana kirjoitettiin C++ ja Python lähdekoodia yli 6000 riviä. Kehitetyn ohjelmiston toteutusteknisiä yksityiskohtia ei käsitellä tässä työssä kovinkaan tarkasti. Ohjelmiston UML-kaaviot löytyvät liitteestä A.5 ja työn aikana tuotettu ohjelmakoodi työn mukana tulevalta CD:ltä (liite A.4), joten toteutuksen yksityiskohdista kiinnostuneet voivat tutustua luokitteluohjelmiston toteutukseen työn liitteisiin tutustumalla.

### 4.1.1 Käytetyt ohjelmistot ja tekniikat

Koska työn yhtenä tavoitteena oli tuottaa konenäkölaadunvalvontaan soveltuva luokittelijaohjelmisto, valittiin toteutustekniikka siten, että työssä toteutettujen ohjelmistokomponenttien integrointi olemassa oleviin sovelluksiin olisi mahdollisimman vaivatonta. Työn tilaajan aikaisempi lähdekoodi on kirjoitettu C++ ohjelmointikielellä, joten se valittiin myös tämän työn toteutuskieleksi.

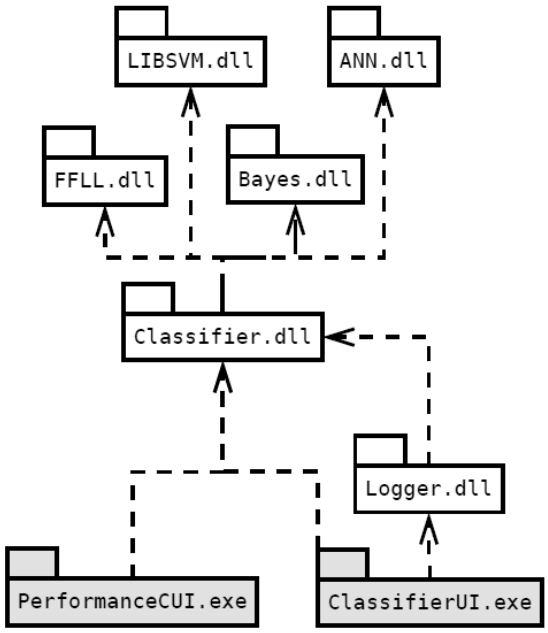
Työssä käytettiin Microsoft Windows XP Professional SP 3 ja Ubuntu Linux 9.04 -käyttöjärjestelmiä ja niiden tarjoamia työkaluja. Luokitteluohjelmiston kehittämiseen käytettiin Microsoft Visual C++ 2008 Express Edition -sovelluskehitintä, versionumeroltaan 9.0 SP1. Graafisen käyttöliittymän sommitteluun käytettiin ResEdit v1.4.8 -ohjelmistoa. Testien automatisointiin käytettiin Python-ohjelmointikielen ja -tulkin versiota 2.6.1. Testiskriptien kirjoittamiseen ja raakadatan tarkasteluun käytettiin NotePad++ v. 5.1.4 -tekstieditoria. Tulosten käsittelyyn käytettiin OpenOffice 3.0.0 Calc -taulukkolaskentaohjelmaa.

Työssä suosittiin vapaasti saatavilla olevia ohjelmistokirjastoja niiden tuoman ajansäästön vuoksi. Tämä aikaisessa vaiheessa tehty päätös käyttää valmista ohjelmistokoodia silloin, kun sitä on saatavilla, mahdollisti monipuolisen luokitteluohjelmiston toteuttamisen. Käytetyt valmiskirjastot on listattu liitteessä A.1.

Piirreilmaisimet eivät olleet tämän työn keskiössä, mutta niiden käyttämiseltä ei voitu välttyä, sillä työn tilaajalta saadusta laadunvalvonnan testiaineistosta ei piirteitä oltu valmiiksi irrotettu. Testiaineiston valmistelemisessa luokittimille sopivaan muotoon käytettiin MVTec Halcon -konenäkökirjaston algoritmeja. Kirjasto tarjoaa suuren määrän erilaisia valmiita työkaluja, joilla virheiden etsintä ja piirreilmaisimet voitiin helposti ja nopeasti toteuttaa.

### 4.1.2 Ohjelmiston rakenne

Työssä kehitetty luokittelijaohjelmisto koostuu luokitinkirjastosta ja kahdesta käyttöliittymästä: testien ajamiseen tarkoitetusta komentorivikäyttöliittymästä sekä datan visualisoimiseen ja luokittimien parametrien säätämiseen tarkoitetusta graafisesta käyttöliittymästä. Näiden lisäksi ohjelmistoon kuuluvat varsinaiset luokitinmenetelmät toteuttavat ohjelmistokirjastot. Ohjelmiston komponenttien riippuvuudet on kuvattu kuvan 4.1 UML-kaaviossa.




***Kuva 4.1.*** *UML-kaavio luokitteluohjelmiston ohjelmistomoduuleista.*

Ohjelmiston ytimenä toimii luokitinkirjasto `Classifier.dll`, joka toteuttaa luokitinpalvelut käyttäen luokitinmenetelmäkirjastoja `ANN.dll`, `Bayes.dll`, `FFLL.dll` ja `LIBSVN.dll`. `ANN.dll` on lähinaapureiden nopean etsinnän toteuttava kirjasto, `Bayes.dll` on naiivin Bayes -luokittimen toteuttava kirjasto, `FFLL.dll` on sumean

säätimen toteuttava kirjasto ja `LIBSVM.dll` SVM -menetelmät toteuttava kirjasto. Näistä `Bayes.dll` kehitettiin työn kuluessa. Valmiskirjastoihin tehdyt muutokset puolestaan käsitellään luvun 4.1.3 menetelmäkohtaisissa aliluvuissa. `Classifier.dll` toimii luokitinkehyksenä, joka mahdollistaa erityyppisten luokitinmenetelmien käyttämisen yhdenmukaisen ohjelmistorajapinnan kautta. Luokitinrajapinnan määrittävään IClassifier-rajapintaluokkaan voi tutustua liitteen A.5 UML-kaaviosta tai työn liitteenä olevan CD levyn lähdekoodeista. CD-levyn sisältöä on esitelty liitteessä A.4. `Classifier.dll`:n vastuulla on luokittelun lisäksi myös datajoukkojen mallinnus. Esimerkki kirjaston käytöstä on liitteessä A.2. Esimerkissä ladataan levyltä piirrevektori-datajoukko, jaetaan se opetus- ja testijoukkoon, opetetaan luokitin opetusjoukon avulla ja luokitellaan testijoukon alkiot.

`Logger.dll` -moduliin on kerätty muutamia luokittimien toiminnan analysointiin tarkoitettuja aputyökaluja, kuten virheluokitusmatriisin (*confusion matrix*) ja virheluokitusprosentin laskeminen.

`ClassifierUI.exe` on datan ja tulosten visualisointiin tarkoitettu Windows-ohjelma. Piirreavaruuden piirtämiseen käytetään OpenGL:ää, joten suurien ja monidimensionaalisten datajoukkojen visualisointi on mahdollista.

`PerformanceCUI.exe` on komentorivikäyttöliittymä, joka mahdollistaa testien automatisoidun ajamisen. Ohjelman ajaminen komentoriviltä `-h` optiolla antaa ohjelman suppean käyttöohjeen. Käytetyt testiohjelmat löytyvät työn mukana tulevalta CD-levyltä.

### 4.1.3 Luokittimien toteutus

Valituista luokittelumenetelmistä luokitteluohjelmistoon toteutettiin neljä päämenetelmää, jotka ovat:

**$k$-NN** Lähinaapuriluokitin

**Fuzzy Classifier** Sumeaan päättelyyn perustuva luokitin

**Naive Bayes** Bayesilaista päättelyä käyttävä tilastollinen luokitin

**SVM** Tukivektorikone (*Support Vector Machines*)

Nämä voidaan jakaa vielä seuraaviin alimenetelmiin:

**1-NN** Lähinaapuriluokitin, jossa luokitus määräytyy lähimmän naapurin mukaan

**3-NN** Lähinaapuriluokitin, jossa etsitään kolme lähintä naapuria

**5-NN** Lähinaapuriluokitin, jossa etsitään viisi lähintä naapuria

**7-NN** Lähinaapuriluokitin, jossa etsitään seitsemän lähintä naapuria

**FT** Sumea luokitin kolmion muotoisilla (*triagular*) sumeilla joukoilla

**FZ** Sumea luokitin puolisuunnikkaan muotoisilla (*trapezoid*) sumeilla joukoilla

**FB** Sumea luokitin kellokäyrän muotoisilla (*bell*) sumeilla joukoilla

**NBN** Naiivi Bayes Gaussin jakaumaestimaateilla

**NBK** Naiivi Bayes ydinmenetelmän jakaumaestimaateilla

**NBD** Naiivi Bayes diskretisoiduilla jakaumaestimaateilla

**CSVML** C-SVM (tyypin 1 SVM) lineaarisella kernelillä

**CSVMR** C-SVM (tyypin 1 SVM) RFB kernelillä

**nuSVML** $\nu$-SVM (tyypin 2 SVM) lineaarisella kernelillä

**nuSVMR** $\nu$-SVM (tyypin 2 SVM) RFB kernelillä

Seuraavaksi esitellään lyhyesti luokitinmenetelmien toteutukset. Esittely on tarkoituksella pidetty hyvin yleisellä tasolla, sillä toteutuksen yksityiskohdista kiinnostuneet voivat tutustua työn mukana toimitettavalta CD:ltä löytyviin lähdekoodeihin. Toteutuksien matemaattiset lähtökohdat on esitelty Teoria-osiossa.

**Sumea luokitin**. Sumean luokittimen toteuttamiseksi käytettiin C++:lla kirjoitettua kirjastoa nimeltä FFLL (*The Free Fuzzy Logic Library*). FFLL perustuu Steve Rabinin toimittamassa kirjassa AI Game Programming Wisdom [57, s. 90-102] esiteltyyn sumean päättelykoneen rakenteeseen. Kirjaston on kehittänyt yritys nimeltä *Louder Than A Bomb! Software* Spark!-tuotettaan varten. Kirjasto on lisensoitu LGLP-lisenssillä [21], joten lisenssi ei rajoita luokitteluohjelmiston jakelua tai kehittämistä. FFLL asetettiin käyttämään luvussa 3.4 esiteltyä sumeaa Mamdani-tyypin IF-THEN -päättelyä.

FFLL ei sellaisenaan soveltunut luokittelijaohjelmistoon, sillä sen rajapinta ei mahdollistanut sumean päättelykoneen rakentamista ohjelmallisesti. Ainoa tapa valmistella sumea päättelykone oli ladata sanalliset muuttujat, sumeat joukot ja sääntökanta tiedostosta. FFLL tukee FCL-tiedostoformaattia (*Fuzzy Control Language*), joka on IEC 61131 standardin [28] mukainen sumean ohjauksen kuvauskieli. Sumean luokittimen toteuttamiseksi FFLL:n rajapintaa laajennettiin tukemaan sumean päättelykoneen elementtien määrittämistä.

Työn kuluessa paljastui, että FFLL -kirjaston sisäisestä toteutuksesta johtuen sumea luokitin ei suoriudu suuridimensionaalisista tai moniluokkaisista ongelmista. Ongelman aiheuttaa kirjaston tapa laatia aina täysi sääntökanta, minkä vuoksi sääntökannan koko kasvaa räjähdysmäisesti piirteiden ja luokkien määrän lisääntyessä. Tämä kombinatorisen räjähdyksen aikaansaama ongelma on ikävä rajoite,

joka olisi ollut ratkaistavissa toteuttamalla sumea päättely paremmin kuin miten se on tehty FFLL:ssä. Oman sumean päättelykoneen ohjelmointiin ei ryhdytty, sillä työhön käytetty aika oli rajallinen ja sumean luokittimen vahvuudet ovat muualla kuin suurten ongelmien mallintamisessa ja ratkaisemisessa. Rajoituksistaan huolimatta FFLL-kirjastoa käyttävä sumea luokitin vastaa sille asetettuja vaatimuksia ja soveltuu niihin tehtäviin, joiden ratkaiseminen sumean sääntökannan avulla on tarkoituksenmukaista.

**Lähinaapuriluokitin**. Lähinaapuriluokittimen toteuttamiseksi käytettiin Mountin ja Aryan kehittämää ANN-ohjelmistokirjastoa. ANN pyrkii korjaamaan perinteisen lähinaapuriluokittimien suuret suoritusaikavaatimukset etsimällä luokiteltavalle piirreavaruuden pisteelle likimääräiset lähimmät naapurit eksaktien naapureiden sijaan. Tämä menetelmä on kuvattu luvussa 3.3. Ohjelmistokirjasto tarjoaa keinot säätää naapureiden etsinnän tarkkuutta, joten luokittelun nopeutta voidaan tarpeen vaatiessa lisätä tarkkuuden kustannuksella.

Luokitusalgoritmin äänestysosio toteutettiin itse käyttäen luvun 3.3 $k$-NN luokittimen määritelmää (3.31). Lisäksi ANN-kirjaston BBD-puun tallennusmuoto käärittiin IEC 61131 standardin [28] mukaista FCL-tiedostoformaattia mukailevaan .clf-tiedostomuotoon.

**Naiivi Bayes -luokitin**. Naiivi Bayes -luokitin toteutettiin ilman kolmannen osapuolen kirjastoja. Varsinaisen päättelykoodin lisäksi toteutettiin kolme erilaista estimointimallia prioritodennäköisyyksille. Toteutettu naiivi Bayes -luokittelu ja prioritodennäköisyyksien estimointimallit on kuvattu luvussa 3.2.

Yksinkertaisin näistä jakaumamalleista olettaa luokka- ja piirrekohtaisten näytteiden olevan normaalijakautuneita. Piirrevektoreille lasketaan luokka- ja piirrekohtaiset keskiarvot ja keskihajonnat, joiden avulla muodostetaan normaalijakaumat. Luokkaa ennustettaessa piirrearvon esiintymisen todennäköisyys lasketaan normaalijakauman tiheysfunktion (3.12) avulla.

Edellistä hienostuneempi menetelmä on luoda opetusjoukon piirrearvojen kohdille pieniä normaalijakaumia ytimiksi. Luokiteltavan näytteen prioritodennäköisyyttä laskettaessa näiden ytimien normaalijakaumien tiheysfunktioiden arvot summataan luokka- ja piirrekohtaisiksi prioritodennäköisyyksiksi. Koska ytimien summaaminen saattaa olla laskennallisesti raskasta, tätä pyrittiin keventämään summaamalla vain ytimet, joille pätee

$$|x_i - \mu_j| < 6\sigma. \tag{4.1}$$

Perusteluna tälle on, että ytimen pinta-alasta vain $0.34 \cdot 10^{-3}\%$ on tämän rajan ulkopuolella [14]. Empiiriset testiajot vahvistivat, ettei optimoinnista ollut haittaa testiaineistojen luokitustarkkuuteen.

Kolmas toteutettu jakauman estimointimenetelmä on teoriaosassa kuvattu diskretisointimenetelmä, jossa jakauma diskretisoidaan käyttäen hyväksi opetusjoukon luokkatunnisteiden jakautumista piirrearvon funktiona. Luokkaentropian avulla lasketaan diskretisoinnin leikkauskohdat. Diskreetit arvot saadaan laskemalla diskretisoidulla välillä olevat opetusjoukon luokitussuhteet.

Opetettuja naiivi Bayes -luokittimia voidaan kirjaston avulla myös ladata ja tallentaa tietokoneen levylle. Tiedostomuotona on .clf päätteinen tekstitiedosto, jonka rakenne mukailee sumean luokittimen toteutuksen yhteydessä esiteltyä IEC 61131 standardin [28] mukaista FCL-tiedostoformaattia.

**SVM-luokitin**. Luokitin toteutettiin käyttäen Changin ja Linin kehittämää ja ylläpitämää LIBSVM kirjastoa. Kirjasto toteuttaa sekä ykköstyypin ($C$-SVC) että kakkostyypin ($\nu$-SMC) SVM-luokittimet. Kirjasto on lisensoitu LGPL-lisenssillä [21], joten sen käyttäminen ei rajoita luokitteluohjelmiston jakelua.

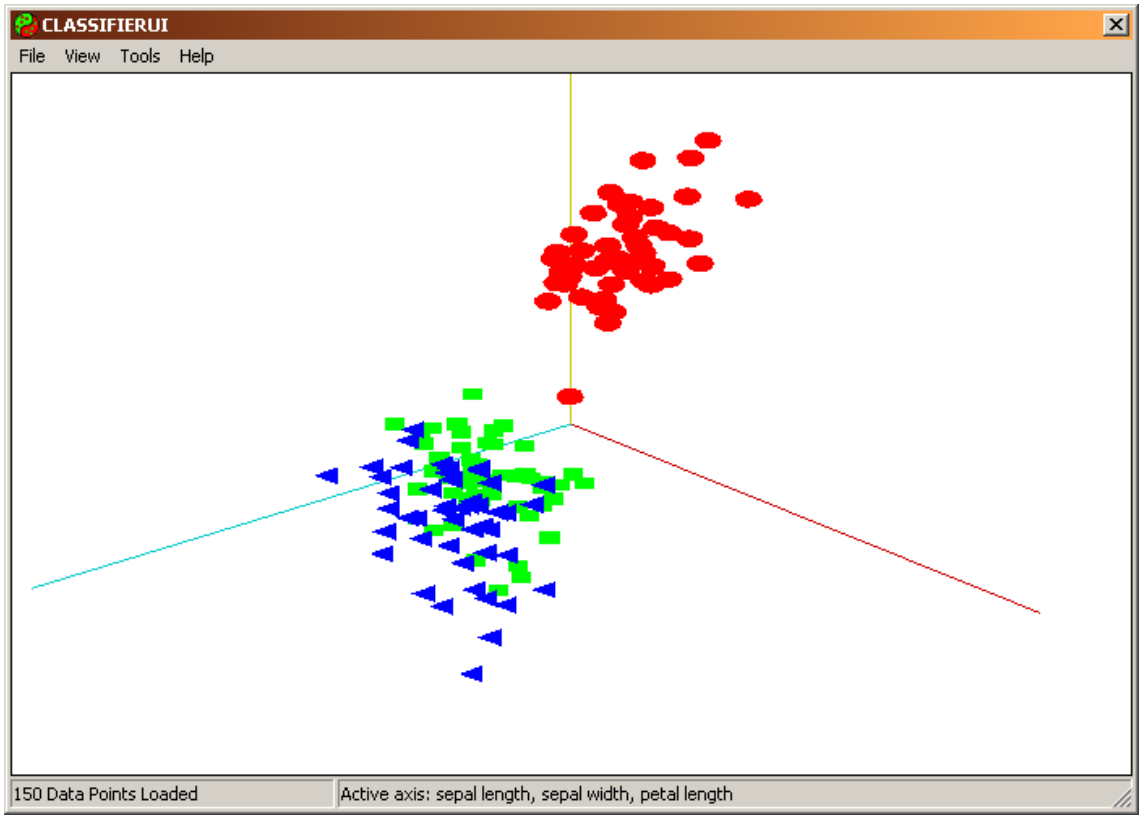
SVM-luokitin on toteutetuista menetelmistä vaikeimmin säädettävä. Toteutuksella on kolme säädettävää parametria, jotka ovat $C$ ja $\nu$ päätöspintojen tarkkuuden määrittämiseksi ($C$ ykköstyypin ja $\nu$ kakkostyypin SVM-luokitinta varten) ja $\gamma$ RBF-kernelin muodon määrittämiseksi. Hsu et al. [25] on hyvä ohje näiden parametrien arvojen haarukointiin.

SVM-luokittimen tallennusmuoto on LIBSVM kirjaston tarjoama tiedostoformaatti SVM-mallien tallentamiseen. Tiedostoformaatti on siis erilainen kuin muilla luokittimilla.

### 4.1.4 Ohjelmiston ominaisuudet

Luokitteluohjelman graafisen käyttöliittymän avulla voi testata luokittimia eri datajoukoilla ja parametreilla ja esittää niiden tuloksia graafisesti. Käyttöliittymä on esitetty kuvassa 4.2. Esimerkissä ohjelmaan on ladattu 150 pisteen opetusaineisto, jonka piirreavaruuden kolme ensimmäistä ulottuvuutta on piirrettynä ruudulle. Pitämällä hiiren painiketta pohjassa ja liikuttamalla hiirtä voi piirreavaruutta kääntää haluamaansa suuntaan. Piirreavaruutta voi myös suurentaa ja pienentää käyttämällä hiiren rullaa.

Ohjelman *File*-valikosta käsin voi ladata ohjelmaan opetusjoukkoja, tallentaa luokiteltuja testijoukkoja sekä tallentaa ja ladata opetettuja luokittimia levyltä. *View*-valikosta voi valita visualisoitavat piirreavaruuden ulottuvuudet sekä näyttää opetusjoukon ja viimeisimmän luokitteluoperaation tiedot. *Tools*-valikon kautta voi valita käytettävän luokittimen ja säätää sen parametreja. Valikosta valitaan myös luokittimen testauksessa käytettävä menetelmä ladatun aineiston jakamiseksi opetus- ja testijoukkoihin. Aineisto voidaan ennen luokittelua normalisoida siten, että kaikki piirteet skaalataan välille $[0, 1]$. Kun menetelmät on valittu ja aineisto on esikäsitelty, voidaan *Tools*-valikon *Train*- ja *Classify*-komennoilla opettaa luokittimia sekä




***Kuva 4.2.*** *Luokittelijaohjelmiston graafinen käyttöliittymä*

luokitella niiden avulla levyltä ladattuja aineistoja.

Parametrien säätämiseen ja niiden vaikutuksen testaamiseen tarvittavien ominaisuuksien lisäksi luokitteluohjelmisto osaa laskea takaisinsijoitus- ja pidätysestimaatit luokitteluvirhetodennäköisyydelle. Lisäksi ohjelmiston avulla voidaan laatia virheluokitusmatriisi (*confusion matrix*), joka kertoo mihin luokkiin kuuluvat luokitukset menivät väärin. Matriisin $(i, j)$:s elementti kertoo mitkä luokkaan $\omega_j$ kuuluvat näytteet luokiteltiin luokkaan $\omega_i$. Näin ollen virheluokitusmatriisin diagonaalilta voidaan lukea oikein tehdyt luokitukset ja diagonaalin ulkopuolelta virheluokitukset virheellisine kohdeluokkineen. Virheluokitusmatriisi kertoo miten virheluokitukset ovat jakaantuneet, joten sen avulla voidaan analysoida uuden piirreilmaisimen tai luokittimen painokertoimien muuttamisen tarvetta.

Komentoriviohjelmisto PerformanceCUI tarjoaa lähes kaikki nämä toiminnot komentorivioptioiden kautta. Vaihtoehtoinen käyttöliittymä mahdollisti työssä tehtyjen testien automatisoinnin.

## 4.2 Aineisto

Esiteltyjen luokitinmenetelmien testaamiseksi toteutetulla ohjelmistolla tarvittiin testiaineistoja. Testiaineistoja kerättiin kirjallisuudesta, konenäköalan suomalaisilta

asiantuntijoilta ja työn tilaajalta. Testiaineistot ovat tarkoituksella eri kokoisia, jotta luokittimien toimintaa erilaisilla näytteiden lukumäärillä, piirteiden lukumäärillä ja kohdeluokkien määrillä voitiin testata.

Kirjallisuuden luokitinvertailuissa käytettyjä testiaineistoja otettiin mukaan varmistamaan, että toteutettujen luokittimien ominaisuudet, kuten luokittelutarkkuus ja suorituskyky vastaavat kirjallisuudessa mainittuja. Konenäkölaadunvalvonta-alan aineistolla puolestaan oli tarkoitus testata luokittimien selviytymistä mahdollisimman todenmukaisista laadunvalvontatehtävistä.

***Taulukko 4.1.*** *Työssä käytetyt testiaineistot*

| **Nimi** | **Näytteitä** | **Piirteitä** | **Luokkia** |
|---|---|---|---|
| ”Iris” | 150 | 4 | 3 |
| ”Glass” | 214 | 9 | 7 |
| ”Metallivalu” | 12 | 4 | 3 |
| ”Paperi” | 626 | 2 (112) | 16 |
| ”Ruiskuvalu” | 213 | 12 | 5 |

Työssä käytetyt aineistot on listattu taulukossa 4.1. ”Iris” ja ”Glass” ovat luokittimien ja koneoppimisen testaamiseen tarkoitettuja testiaineistoja. ”Metallivalu”, ”Paperi” ja ”Ruiskuvalu” ovat konenäkölaadunvalvonta-alan aineistoja, joissa luokitellaan visuaalisia virheitä eri virheluokkiin. Aineistot ja niiden ominaisuudet on esitelty tarkemmin seuraavissa aliluvuissa.

### 4.2.1 Kirjallisuuden testiaineistot

Kirjallisuudessa esiintyvistä tieteellisten luokitintestitapausten joukosta valittiin kaksi datajoukkoa, joita käytettiin alustavaan luokittimien testaamiseen. Testijoukkojen avulla voitiin verrata toteutettujen luokitinmenetelmien suorituskykyä kirjallisuudessa esiintyviin. Näin saatiin varmuus siitä, että luokittimet toimivat ja että niiden suorituskyky vastaa kirjallisuudesta löytyviä lukuja.

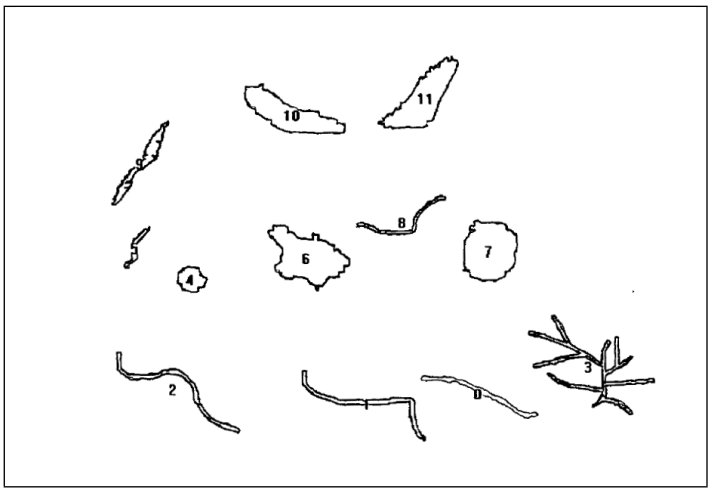
Kirjallisuuden testidatajoukot valittiin myös sen mukaan, kuinka läheisesti ne muistuttivat konenäkölaadunvalvonnan piirreluokittelutehtävissä esiintyviä datajoukkoja. Kirjallisuuden testiaineistossa esiintyviltä piirteiltä edellytettiin samankaltaisuutta konenäkölaadunvalvonnassa esiintyviin piirremittoihin.

**”Iris”**. Ensimmäinen testidatajoukko on hahmontunnistuskirjallisuuden klassikko ”Iris”, jonka Fisher [20] esitteli vuonna 1936 diskriminanttianalyysin esimerkkinä. Datajoukkoon on mitattu 150 kukkayksilön terälehtien pituuksia ja leveyksiä. Luokittelutehtävänä on mittojen perusteella päätellä kukkayksilön lajike. Piirteitä datajoukossa on 4, luokkia (eli lajikkeita) 3.

**”Glass”**. Toiseksi tieteellisten testien datajoukoksi valittiin tarkoituksella haastavampi luokittelutapaus. Datajoukon nimi on ”Glass” ja se koostuu 214 lasinäytteestä,

joiden reflektiivisyys ja oksidipitoisuudet (yhteensä 9 eri tunnuslukua) tunnetaan. Luokkia (eli lasityyppejä) on 7 erilaista. Datajoukko on UCI Machine Learning Repository -kokoelmasta [65].

### 4.2.2 Laadunvalvonta-alan testiaineistot




***Kuva 4.3.*** *Metallivalun visuaalisia virheitä [71]*

**”Metallivalu”**. Ensimmäinen laadunvalvonta-alan aineisto on Wong et al. [71] esittelemä 12:n käsingeneroidun visuaalisen virheen datajoukko (kuva 4.3). Tässä työssä siitä käytetään nimeä ”Metallivalu”. Pienestä koostaan ja keinotekoisuudestaan huolimatta se tuli valituksi, sillä se oli ainoa visuaalisten virheiden luokittelun testaamiseen tarkoitettu datajoukko, joka kirjallisuuskartoituksen aikana kirjallisuudesta löytyi. Tutkimuksessaan Wong et al. ovat käyttäneet seuraavaa neljää koko- ja asentoinvarianttia piirremittaa metallivalujen visuaalisten virheiden luokitteluun:

- Halkaisijan suhdeluku *Radius Ratio*, $RR = \frac{d_{min}}{d_{max}}$, missä $d_{min}$ ja $d_{max}$ ovat pienin ja suurin alueen keskipisteen läpi mitattava halkaisija.
- Akseleiden suhdeluku *Axis Ratio*, $AR$ = Alueen ympäri sovitetun ellipsin isoakselin ja pikkuakselin suhde.
- Likimääräinen pinta-ala *Approximate Area*, $AA = \frac{A_o}{A_r}$, missä $A_o$ on alueen pinta-ala ja $A_r$ on pienimmän alueen sisäänsä sulkevan nelikulmion pinta-ala.
- Reunaviivan suhde, $PPDA = \frac{P^2}{A_o}$, missä $P$ on reunaviivan pituus ja $A_o$ on alueen pinta-ala.

”Metallivalu”-testiaineistossa on kolme luokitusta virheille: *halkeama*, *reikä* ja *repeymä*. Edellä kuvattujen neljän suhteellisen piirteen lisäksi käytettävissä on myös

niiden alkuarvot: pienin ja suurin virhealueen keskipisteen läpi mitattava halkaisija, virhealueen ympäri sovitetun ellipsin isoakselin ja pikkuakselin mitat, virhealueen reunaviivan pituus sekä virhealueen pinta-ala. Yhteensä piirteitä on siis 10, joista kuusi on virheistä mitattuja piirrearvoja ja loput neljä mittaustuloksista johdettuja asento- ja kokoinvariantteja piirteitä. Luokittelussa käytämme johdettuja piirteitä.

**”Paperi”**. Toinen visuaalisten virheiden luokittelua testaava aineisto saatiin Jyväskylän Yliopiston konenäköryhmältä. Kyseessä on paperin valmistamisessa havaittujen visuaalisten virheiden luokittelutehtävä, jossa virhenäytteitä on 626 kappaletta ja virheluokituksia 16. Virheitä kuvaavat piirteet ovat harmaasävyhistogrammi ja Tirrosen et al. [64] esittelemät usean asteen gradienttihistogrammit (MOGH, *multiple order gradient histogram*).

***Taulukko 4.2.*** *Työn tilaajan virhetyyppimäärittelyt tarkastettavan kappaleen kirkkaalle alueelle. Löydettävyystaso (Taso) on ilmaistu asteikolla 0-5 (0 = todella vaikea löytää, 5 = löytyy helposti)*

| **Nimi** | **Kuvaus** | **Taso** |
|---|---|---|
| Foiliroska | Kappaleeseen on jäänyt foilin palanen, mitä ei saa pyyhkimällä pois ja näkyy yleensä isona virheenä. | 5 |
| Imukupin jälki | Harmaa rengas kappaleen pinnalla, virhettä ei aina näe suoraan katsottaessa, mutta näkyy peilaamalla | 1 |
| Kirkas piste | Kirkas painauma kappaleen pinnalla, mikä syntyy esimerkiksi kun muotin välissä on likaa. | 4 |
| Imujälki | Painauma kappaleen pinnalla. | 3 |
| Musta piste | Musta piste, joka on syntynyt esimerkiksi palaneesta raaka-aineesta. | 5 |
| Naarmu | Viiva kappaleen pinnalla. Syntynyt luultavasti kappaleenkäsittelyssä. | 4 |
| Virtausjälki | Harmaa jälki kappaleessa, mikä näkyy ainoastaan kääntelemällä kappaletta saaden valon peilauksen. | 0 |
| Pesujälki | Harmaa tasainen ohut kalvo tai harmaa tasainen piste. | 1 |
| Öljyläikkä | Harmaa tasainen ohut kalvo tai harmaa tasainen piste. | 1 |

**”Ruiskuvalu”**. Työn tilaajalta saatiin yksi luokittimen testaamiseen sopiva testiaineisto, joka oli joukko ohjauspaneelin näytön linssin visuaalisten virheiden tarkistuksen aikana kerättyjä kuvia. Tarkastustehtävänä aineistossa oli visuaalisten virheiden löytäminen ruiskuvalettujen kappaleiden näyttöalueelta. Tämän tyyppisillä tarkastettavilla pinnoilla esiintyvät virhetyypit on esitelty taulukossa 4.2. Tässä työssä luokittelemme virhetyyppejä jotka oli mahdollista löytää kuva-aineistosta suhteellisen vaivattomasti. Tämä tarkoitti löydettyjen virheiden luokittelua virhetyyppeihin foiliroska, kirkas piste, imujälki, musta piste ja naarmu.

Aineisto saatiin työn tilaajalta jalostamattomassa bittikarttamuodossa, joten työssä tehtiin tälle aineistolle myös kuvankäsittely, segmentointi ja piirteiden irrotus.

Tähän käytettiin Halcon konenäkökirjastoa, joka tarjoaa joukon valmiita konenäköalgoritmeja ja piirreilmaisimia. Kirjaston avulla aineistoille laskettiin seuraavat kaksitoista piirrettä:

**Harmaasävyt** - Virheeksi tunnistetun alueen keskimääräinen harmaasävy ja harmaasävyjen keskihajonta. Näitä piirrearvopareja on kaksi, sillä jokaisesta kappaleesta on kaksi eri valaistuksella otettua kuvaa, joista molemmille on laskettu harmaasävypiirteet.

**Pinta-ala** - Virheeksi tunnistetun alueen pikseleiden lukumäärä, jota merkitään tunnisteella $A_o$.

**Konveksisuus** - Tunnusluku joka ilmoitetaan suhdelukuna $C_v = \frac{A_o}{A_{CH}}$, missä $A_{CH}$ on konveksin verhon (*convex hull*) pinta-ala. $C_v < 1$ jos virhe on konkaavi alue tai siinä on reikiä.

**Pyöreys** - Piirre kertoo virheen samankaltaisuuden ympyrän kanssa. Se saadaan kaavasta $C_c = \frac{A_o}{{r_{max}}^2\pi}$, missä $r_{max}$ on suurin reunaviivan ja massakeskipisteen välinen etäisyys.

**Reunaviivan pituus** - Tunnusluku $L$ lasketaan kulkemalla reunaviivan muodostavien pikseleiden kautta niin, että akseleiden suuntaiset siirtymät kasvattavat reunaviivan pituutta yhdellä ja viistot siirtymät $\sqrt{2}$:lla.

**Kompaktisuus** - Jos $L$ on reunaviivan pituus ja $A_o$ virhealueen pinta-ala, niin kompaktisuus voidaan määritellään kaavan $C_o = \frac{L^2}{4A_o\pi}$ mukaisesti.

**Suorakulmaisuus** - Tunnusluvun laskemiseksi lasketaan suorakulmio, jonka ensimmäinen ja toinen momentti vastaavat löydetyn virhealueen momentteja. Tämän jälkeen tunnusluku saadaan kaavasta $C_r = \frac{A_d}{A_r}$, missä $A_d$ on sen alueen pinta-ala, joka kuuluu sekä suorakulmioon, että alkuperäiseen alueeseen ja $A_r$ on suorakulmion pinta-ala. Suorakulmion muotoisille kappaleille piirrearvo on 1.

**Pienimmän sovitetun ympyrän säde** - $r_s$, joka on pienimmän virhealueen ympärille sovitetun ympyrän säde.

**Suurimman sovitetun ympyrän säde** - $r_i$, joka on suurimman virhealueen sisäpuolelle sovitetun ympyrän säde.

Tilaajan testiaineistosta löytyi 213 selkeää visuaalista virhettä, jotka luokiteltiin käsin viiteen edellä mainittuun luokkaan. Luokitukset jakaantuivat seuraavasti: roskat $10,3\%$, kirkkaat pisteet $3,8\%$, imujäljet $2,8\%$, mustat pisteet $7,5\%$ ja naarmut $75,6\%$.

## 4.3 Suoritettujen testien kuvaus

Suoritetut testit suunniteltiin siten, että niistä saatavien tulosten avulla voitiin kertoa mikä luokittelumenetelmä tai -menetelmät sopivat parhaiten konenäkölaadunvalvonnassa käytettäväksi. Toteutettujen luokittimien suorituskykyä vertailtiin aineistokohtaisesti, sillä oletuksena oli, että luokittimien soveltuvuudessa on aineistokohtaisia eroja. Luokittimien erot etsittiin vertailemalla virheluokitusten yleisyyttä ja laskennallisia suoritusaikavaatimuksia.

Luokittimia päätettiin opettaa kahdella eri kokoisella testiaineiston osajoukolla. Tilaajan asiantuntijan arvio oli, että tyypillisesti virhemalleja on käytettävissä noin 10 kappaletta. Tätä opetusjoukon kokoa käytettiin mitattaessa luokittimen opetettavuutta ja oppimiskykyä (luku 2.5). Asiantuntija arvioi myös, että parhaimmillaan virhemalleja on saatavilla joitain kymmeniä, minkä vuoksi luokittelutarkkuuden testaamiseksi päädyttiin käyttämään 50 virheen opetusjoukkoa.

Luokittelutarkkuuden mittaukset suoritettiin siten, että kaikille testiaineisto - luokitin yhdistelmille ajettiin luvussa 3.6 esitelty standardi ristiinvalidointitesti. Opetettavuuden testaamiseksi ristiinvalidointitestissä käytetyn aineiston koko oli 11 virhenäytettä. Näin joka ristiinvalidointikierroksella opetusjoukko oli 10 virhenäytettä. Vastaavasti luokittelutarkkuuden testaamiseksi aineistossa oli 51 näytettä. Jokainen testi ajettiin 10000 kertaa, joista laskettiin keskimääräinen virheluokitustodennäköisyys ja virheluokitustodennäköisyyden keskihajonta. Ristiinvalidointitestin opetusaineisto valittiin joka iteraatiolle satunnaisesti koko opetusaineistosta. Yhteensä kutakin luokitin-aineisto yhdistelmää kohden tehtiin opetettavuuden testaamiseksi 110000 opetusta ja luokitusta ja luokittelutarkkuuden testaamiseksi vastaavasti 510000 opetusta ja luokitusta. Molempien operaatioiden kokonaiskesto mitattiin *Time Stamp Counter*(TSC) -ajoitusmenetelmällä, joka pystyy mikrosekunnin tarkkuuksiin [34].

Testit ajettiin kolmella identtisellä työasemalla. Koneiden käyttöjärjestelmä oli Microsoft Windows XP Professional Service Pack 3, prosessorit olivat 3 GHz kaksiytimisiä Intel Core2 Duo E8400 siruja ja keskusmuistia koneissa oli $3,21$ Gigatavua.

# 5. TULOKSET

Tieteellisille testiaineistoille ajettujen testien tarkoitus tässä työssä oli antaa varmuus sille, että luokittimet toimivat. Ajamalla kirjallisuudessa usein käytetty 10-kertainen ristiinvalidointitesti ”Iris” ja ”Glass” -testiaineistoille kymmenen kertaa ($10 \times 10$ *cross validation*) saatiin vertailukelpoisia estimaatteja virheluokitustodennäköisyyksille. Tieteellisille testiaineistoille ajettujen ristiinvalidointitestien tulokset on kerätty taulukkoon 5.1, mihin on kirjattu 10-kertaisten ristiinvalidointitestien luokitusvirheprosenttien keskiarvot ja niiden keskihajonnat.

***Taulukko 5.1.*** *Luokitteluvirhetestien tulokset kirjallisuusvertailua varten. Taulukkoon on merkitty virheluokitusprosentit ja niiden keskihajonnat.*

| **Luokitin** | **Iris** | **Glass** |
|---|---|---|
| 1-NN | 4.7(0.44) | 30.7(1.16) |
| 3-NN | 5.0(0.35) | 29.2(1.39) |
| 5-NN | 4.4(0.47) | 31.3(1.15) |
| 7-NN | 3.6(0.47) | 35.5(1.39) |
| | | |
| FT | 8.2(0.45) | 53.8(1.40) |
| FZ | 13.0(1.01) | 50.0(1.85) |
| FB | 7.8(0.55) | 52.0(1.51) |
| | | |
| NBN | 4.7(0.38) | 54.9(2.25) |
| NBK | 4.8(0.69) | 43.0(2.30) |
| NBD | 8.0(1.33) | 38.0(1.70) |
| | | |
| CSVML | 3.3(0.77) | 43.5(1.67) |
| CSVMR | 3.7(0.65) | 30.5(1.05) |
| nuSVML | 4.1(0.80) | 54.8(4.28) |
| nuSVMR | 4.5(0.71) | 40.5(5.52) |

Vertaamalla taulukon 5.1 tuloksia kirjallisuudessa esiintyviin testituloksiin voimme varmistua siitä, että luokittimet toimivat sillä luotettavuudella, jota menetelmältä voidaan odottaa. $k$-NN menetelmän virheluokitusprosentit vastaavat Vlachos et al. [68] raportoimia tuloksia. Vaikka sumeiden luokittimien opetusmenetelmä onkin yksinkertainen, päästään sillä samoihin tai hieman parempiin luokitustarkkuuksiin kuin mitä Cordón et al. [11] ovat raportoineet sumeita luokittelumenetelmiä käsittelevässä vertailussaan (tässä työssä käytetty Mamadani-tyypin IF-THEN -luokitin vastaa raportin a-tyypin luokitinta). Naiivi Bayes -luokitinvarianttien tu-

lokset vastaavat ”Iris”-aineiston osalta Bouckaertin [3] raportoimia tuloksia. Bouckaert on saanut ”Glass”-aineiston luokittelusta NBN ja NBD -menetelmillä hieman parempia tuloksia, mitä tässä työssä mitattiin. Tämä saattaa selittyä virheluokitustodennäköisyyden estimoinnin epätarkkuuksilla, sillä Bouckaertin raportoimat virheluokitustodennäköisyyden keskihajonnat ovat suhteellisen suuria. Hsu ja Lin [24] ovat saaneet tukivektorikoneiden luokitusmenetelmiä vertaillessaan samansuuntaisia tuloksia kuin mitä tässä työssä mitattiin. Tässä työssä taulukoidut aavistuksen heikommat tulokset selittyvät sillä, että Hsu ja Lin ovat testeissään optimoineet $C$ ja $\gamma$ parametreja, kun taas tässä työssä ne pidettiin opetuksen aikana oletusarvoissaan. Kaikkien työssä toteutettujen menetelmien luokittelutarkkuus vastaa hyvin kirjallisuudessa raportoituja luokittelutarkkuuksia. Tämä indikoi sitä, että luokittimet toimivat siten kuin teoriaosassa on esitelty.

Luvussa 4.3 kuvatut testit suoritettiin ajamalla opetettavuutta mittaava ristiinvalidointitesti kaikille aineisto-luvussa 4.2 kuvatuille datajoukoille. Luokitustarkkuutta mittaavat testit ajettiin kaikille aineistoille 'Metallivalu”-aineistoa lukuunottamatta, jossa ei ollut riittävästi virhenäytteitä testin suorittamiseksi. Testien avulla lasketut keskimääräiset virheluokitusprosentit (*mean error rate*) ja virheluokitusprosentin keskihajonnat on kirjattu taulukoihin 5.2 ja 5.3. Tunnuslukujen avulla voidaan arvioida luokittimien suorituskykyä eri opetusdatajoukon koon, luokkien ja piirteiden määrän mukaan.

***Taulukko 5.2.*** *Luokitteluvirhetestien tulokset opetettavuuden mittaamista varten. Taulukkoon on merkitty virheluokitusprosentit ja niiden keskihajonnat.*

| **Luokitin** | **Iris** | **Glass** | **Metalliv.** | **Paperi** | **Ruiskuv.** |
|---|---|---|---|---|---|
| 1-NN | 11.5(10.3) | 54.3(18.4) | 18.20(5.3) | 64.8(17.7) | 23.2(14.7) |
| 3-NN | 19.5(12.3) | 57.5(18.4) | 33.3(9.3) | 66.5(17.0) | 22.5(12.7) |
| 5-NN | 32.2(16.8) | 62.0(19.0) | 33.3(6.8) | 70.3(17.0) | 24.3(13.3) |
| 7-NN | 44.7(20.5) | 64.6(20.6) | 62.8(17.1) | 73.3(17.4) | 24.6(13.3) |
| FT | 30.2(10.2) | 84.4(14.6) | 33.4(5.7)* | 84.9(14.2)** | 53.5(15.5) |
| FZ | 29.6(10.6) | 88.3(15.5) | 24.2(5.6)* | 87.8(15.2)** | 50.0(19.1) |
| FB | 30.9(9.8) | 97.6(7.7) | 35.7(4.4)* | 91.5(14.5)** | 39.6(17.7) |
| NBN | 29.7(13.1) | 88.4(17.3) | 31.8(7.0) | 98.3(6.2) | 28.7(16.1) |
| NBK | 43.3(21.9) | 66.4(22.1) | 51.6(11.4) | 75.6(19.6) | 24.8(13.2) |
| NBD | 26.7(13.0) | 71.1(19.5) | 26.7(5.8) | 98.1(5.3) | 34.3(18.9) |
| CSVML | 26.7(16.7) | 60.2(22.7) | 27.8(6.8) | 67.6(18.8) | 23.2(13.6) |
| CSVMR | 16.3(12.4) | 57.9(21.5) | 30.3(10.0) | 68.4(18.2) | **21.2**(**13.3**) |
| nuSVML | **10.5**(**10.1**) | 55.0(18.8) | 19.0(5.8) | **63.7**(**17.9**) | 36.8(29.9) |
| nuSVMR | 10.7(10.3) | **53.8**(**18.5**) | **17.4**(**5.9**) | 67.5(18.0) | 34.0(30.0) |

Taulukossa 5.2 yhdellä tähdellä (*) merkityt virheluokitusprosentit ovat sumean luokittimen yksinkertaisen visuaalisten virheiden luokittelutehtävän tuloksia, joita

oli helppo parantaa rakentamalla sääntökanta käsin. Käsin rakennetulla sääntökannalla kaikki aineiston 12 näytettä luokiteltiin oikein (virheluokitusprosentti 0%). Käytetty sumean päättelyn ohjaustiedosto löytyy liitteestä A.6. Tätä tulosta ei kuitenkaan otettu mukaan vertailuun sen ihmiskomponentin vuoksi. Kahdella tähdellä (**) merkityt virheluokitusprosentit tehtiin karsituille piirrevektoreille, sillä sumea luokitin ei selviytynyt ”Paperi”-aineiston 112-dimensionaalisesta piirreavaruudesta. Täyden piirrevektorin sijaan käytettiin vektoria, jossa oli kaksi piirrettä: kohteen keskimääräinen harmaasävy ja kohteen harmaasävyarvojen keskihajonta.

***Taulukko 5.3.*** *Luokitteluvirhetestien tulokset luokittelutarkkuuden mittaamista varten. Taulukkoon on merkitty virheluokitusprosentit ja niiden keskihajonnat.*

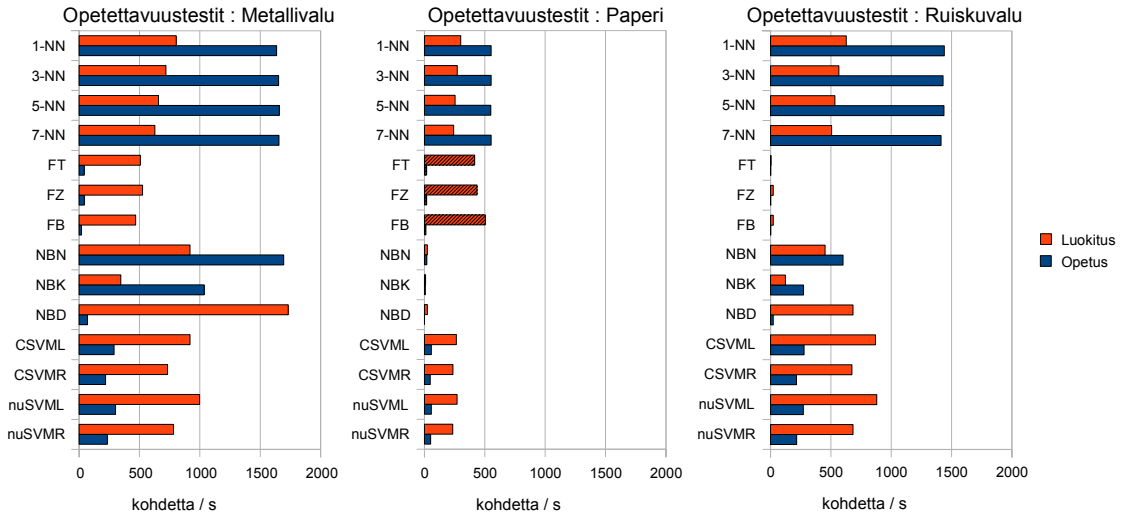
| **Luokitin** | **Iris** | **Glass** | **Paperi** | **Ruiskuv.** |
|---|---|---|---|---|
| 1-NN | 5.5(3.4) | **39.0**(**7.7**) | 51.7(8.4) | 16.3(5.8) |
| 3-NN | 4.7(2.9) | 40.0(7.5) | 50.7(8.2) | 15.3(5.2) |
| 5-NN | 4.7(2.9) | 41.7(7.6) | 50.7(7.8) | 15.9(4.9) |
| 7-NN | 5.1(2.8) | 43.6(7.9) | 52.0(7.6) | 17.1(4.8) |
| FT | 8.7(3.2) | 62.7(10.2) | 80.5(11.3) ∗∗ | 37.3(15.7) |
| FZ | 9.2(3.2) | 65.3(13.3) | 83.7(12.6) ∗∗ | 38.1(15.4) |
| FB | 8.6(3.4) | 70.4(15.3) | 80.8(11.4) ∗∗ | 34.8(14.3) |
| NBN | 5.1(2.7) | 59.5(12.0) | 79.2(11.5) | 40.2(20.3) |
| NBK | 11.2(3.6) | 60.4(10.5) | 72.8(9.1) | 25.6(5.2) |
| NBD | 9.6(3.7) | 55.3(12.2) | 70.2(12.1) ∗∗ | 23.0(7.1) |
| CSVML | 5.2(2.8) | 51.7(10.3) | 50.5(8.3) | 18.5(5.7) |
| CSVMR | 4.3(2.7) | 45.4(10.7) | 50.6(8.1) | **14.4**(**5.0**) |
| nuSVML | **4.1**(**2.9**) | 51.4(10.7) | **48.3**(**8.2**) | 22.0(12.3) |
| nuSVMR | 4.3(3.1) | 40.2(8.4) | 50.7(8.1) | 15.3(6.0) |

Taulukoista 5.2 ja 5.3 voidaan lukea, miten eri luokitinmenetelmät suoriutuvat testiaineistosta oppimisesta ja niiden luokittelusta. Odotetusti SVM-menetelmät suoriutuivat parhaiten lähes kaikissa luokittelutehtävissä. Erityisesti $\nu$-SVM -luokitin RBF-ytimellä on hyvä ja ennen kaikkea johdonmukainen jo pienelläkin opetusdatajoukolla. Lähinaapuriluokitin toimii niin ikään hyvin niin pienillä kun suurillakin opetusjoukoilla. Pienillä opetusjoukoilla kannattaa etsiä vain lähin naapuri, eli käyttää 1-NN menetelmää. Kun opetusjoukon koko kasvaa riittävän suureksi, alkaa 3-NN luokittimella saada 1-NN:ää parempia tuloksia. Kolmea suuremmista $k$:n arvoista oli ainoastaan haittaa testattuja aineistoja luokitellessa. Naiivi Bayes -luokitin pärjää hyvin ”Ruiskuvalu”-aineiston luokittelussa, missä se hyötyy virhetyyppien esiintymistiheyden tuntemuksesta. Bayes-luokittelun tuloksista näkee selvästi, miten parhaiten toimiva jakaumaestimaatiomenetelmä (normaali-, ydinsummaus- tai diskreettijakauma) riippuu aineiston piirteiden todellisista jakaumista, sillä eri estimointimenetelmät ovat vahvoilla eri aineistoilla. Huonoiten naiivi Bayes ja Sumea

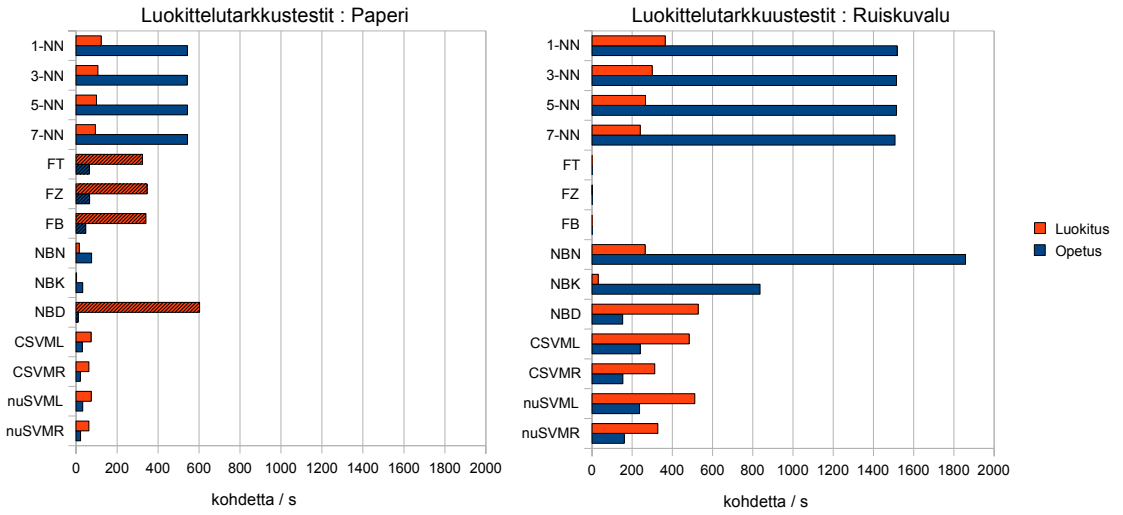
luokitin toimivat kuitenkin silloin, kun piirteiden arvot eivät ole normaalijakautuneita. Sumea luokittelija menestyy opetusmenetelmän heikkouksien vuoksi huonosti vaikeissa luokittelutehtävissä. Sumea luokittelu on kuitenkin 3-NN, NB ja CSVM luokittimia parempi metallivalun visuaalisten virheiden luokittelussa häviten ainoastaan $\nu$-SVM ja 1-NN luokittimille. Suhteellisen hyvä luokittelutarkkuus yhdistettynä sumean luokittimen käsinopetusmahdollisuuteen tekee siitä hyvin soveltuvan menetelmän visuaalisten virheiden luokitteluun.

Tässä työssä esitellyllä menetelmillä ei laadunvalvonta-alan aineiston osalta ylletty asetettujen tavoitteiden mukaisiin virheluokitusprosentteihin. Vika ei kuitenkaan ole luokittelumenetelmissä, sillä ”Iris”-aineiston suorituskykytestit antavat riittävän hyviä tuloksia. Tämä implikoi sitä, että mikäli löydetään riittävä määrä riittävän selvästi luokat erottelevia piirteitä, päästään asetetun tavoitteen mukaiseen virheluokittelutarkkuuteen. Piirteiden valinnan lisäksi kirjallisuudessa on myös muita keinoja luokittelutarkkuuden parantamiseen. Piirteiden valintaa ja muita parannusehdotuksia käsitellään luvussa 6. ”Paperi” -aineisto osoittaa, että osa laadunvalvonnan luokittelutehtävistä on niin vaikeita, ettei täysin luotettavia luokittelupäätöksiä aina voida tehdä.

Opetettavuustestit osoittivat, että kymmenen virhemallia ei riitä luokittimen opettamiseen. Edes 50 virhemallia ei riittänyt laadunvalvonta-alan aineistojen tapauksessa riittävän virheluokitusprosentin saavuttamiseen. Huomata tosin kannattaa, että luokittelutarkkuus olisi todellisuudessa parempi, sillä satunnaisesti valittujen virhemallien sijaan voitaisiin valita hyvin luokkaansa kuvaavia prototyyppejä. 10 opetusnäytettä on silti aineiston tilastolliseen analyysiin perustuville menetelmille liian vähän.




***Kuva 5.1.*** *Luokittimien oppimis- ja luokittelunopeudet opetettavuustesteissä. Pidempi pylväs tarkoittaa parempaa suorituskykyä.*




***Kuva 5.2.*** *Luokittimien oppimis- ja luokittelunopeudet luokittelutarkkuustesteissä. Pidempi pylväs tarkoittaa parempaa suorituskykyä.*

Kuvista 5.1 ja 5.2 on luettavissa luokittimien suorituskyky eri aineistoilla. Sumeiden luokittimien ja NBD -luokittimen lukemat eivät ole täysin vertailukelpoisia muiden kanssa ”Paperi”-aineiston osalta, sillä niiden opettamiseen ja testaamiseen käytettiin muita menetelmiä pienempidimensionaalista lähtödataa. Nämä suorituskykymitat on merkitty kuviin kyseiset palkit viivoittamalla. Kuvan 5.1 opetettavuusteissä opetusjoukon koko oli 10 näytettä ja kuvan 5.2 luokittelutarkkuustesteissä 50 näytettä.

Kuvista voi tehdä huomioita menetelmien välisistä suorituskykyeroista. Lähinaapurietsinnän BBD-kiihdytysmenetelmä näyttää toimivan hyvin, sillä $k$-NN -luokittimen luokitusnopeus on suhteellisen hyvä molemmissa testeissä kaikilla aineistoilla. Yleisesti nopein luokittelumenetelmä on kuitenkin SVM. RBF-kernelin käyttö hidastaa sitä hieman, mutta lineaariseen kerneliin vaihtamalla on mahdollista saada luokittelutarkkuuden kustannuksella luokittelu hieman nopeammaksi. NBD-luokitin on myös todella nopea luokittelija, kunhan piirrevektorin dimensio ei kasva liian suureksi. Mikäli opetuksen nopeus on tärkeää, valinta tehdään $k$-NN -luokittimen ja normaalijakaumia käyttävän naiivin Bayes -luokittimen välillä. Sumeiden luokittimien suorituskyky romahtaa luokkien määrän ja piirreavaruuden dimension kasvaessa. Niiden heikko menestys johtuu kuitenkin ainoastaan käytetyn ohjelmistokirjaston rajoitteista. Paremmin toteutettu ohjelmistokirjasto sumealle luokittelumenetelmälle moninkertaistaisi oppimis- ja luokittelunopeuden ja toisi ne samalle tasolle NBD-menetelmän kanssa.

Kaikki testatut menetelmät ovat riittävän nopeita täyttääkseen luvussa 2.5 asetetut vaatimukset luokittelunopeudelle metallivaluvirheiden luokittelun tyyppisissä tehtävissä. Ruiskuvalettujen kappeleiden visuaalisten virheiden luokittelusta selviä-

vät kaikki muut menetelmät paitsi FFLL -kirjaston rampauttamat sumeat luokittimet. Opetusjoukon koon kasvaessa myös ydinsummajakaumia käyttävä naiivi Bayes -luokitin saattaa osoittautua liian hitaaksi. Suuridimensionaalinen ja moniluokkainen paperin visuaalisten virheiden luokittelutehtävä oli tässä työssä testatuista haastavin. Vain $k$-NN ja SVM -luokittimet pysyisivät tuotannon vauhdissa.

***Taulukko 5.4.*** *Luokittelumenetelmien soveltuvuus konenäkölaadunvalvonnan luokittelutehtäviin*

| **Luokitin** | **Käyt.** | **Yleist.** | **Tark.** | **Nop.** | **Soveltuvuus** |
|---|---|---|---|---|---|
| $k$-NN | 5 | 4 | 4 | 4 | **4.25** |
| FCC | 4 | 1 | 1 | 1 | **1.75** |
| NBN | 4 | 3 | 2 | 3 | **3.00** |
| NBK | 4 | 2 | 1 | 2 | **2.25** |
| NBD | 4 | 3 | 3 | 5 | **3.75** |
| SVM | 3 | 5 | 5 | 4 | **4.25** |

Mitattujen ominaisuuksien perusteella voidaan arvioida luokittimien soveltuvuus konenäkölaadunvalvonnan luokittelutehtäviin. Vertailua helpottamaan keskeiset ominaisuudet on koottu taulukkoon 5.4. SVM ja $k$-NN saavat vertailussa parhaat arvosanat. Taulukon laadintaan käytetty laskentatapa ja sarakkeiden selitykset on esitelty alla.

Taulukon 5.4 arvosanoista **Käytettävyydellä** tarkoitetaan menetelmän intuitiivisuuden ja parametrien määrän yhteisvaikutuksesta syntyvää menetelmän yleistä käyttämisen helppoutta. Se laskettiin kaavalla:

$$\text{Käytettävyys} = \frac{(5 - \text{Parametrien määrä}) + \text{Ymmärrettävyys}}{2}, \tag{5.1}$$

johon ymmärrettävyyden arvot saatiin Michie et al. [45] kokoamasta luokitinvertailusta, jossa arvosana 5 tarkoittaa, että luokittimen toimintaperiaate on helppo ymmärtää ja 1, että se on hyvin vaikea ymmärtää.

**Yleistyvyys** arvosana määräytyi opetettavuuden testaamiseksi tehtyjen testien tulosten perusteella. Näissä testeissä opetusjoukon koko, 10 näytettä, asetti vaatimuksia menetelmien yleistämiskyvylle. Parhaiten yleistyvä menetelmä sai arvosanan 5, seuraava 4 ja niin edelleen. Mikäli menetelmien tulokset olivat todella lähellä toisiaan ne saattoivat jakaa arvosanan.

**Tarkkuus** saatiin tarkkuutta mittaavien testien tulosten perusteella. Opetusjoukon koko oli näissä testeissä 50 näytettä, jonka jo pitäisi riittää luotettavien virheluokitusten tekemiseen. Arvosanat määräytyivät samaan tapaan kuin yleistyvyydelle.

**Nopeus** arvosana määräytyi opetettavuus- ja luokittelutarkkuustesteistä tehtyjen suoritusaikamittausten perusteella (kuvat 5.1 ja 5.2).

**Soveltuvuus** sarake kertoo arvosanan luokittelumenetelmän soveltuvuudelle konenäkölaadunvalvonnan tehtäviin.

Luvussa 4.3 kuvattujen testien lisäksi ajettiin joukko kattavia testejä, jolla tutkittiin opetusjoukon koon vaikutusta luokittelun luotettavuuteen. Testeissä kasvatettiin opetusjoukon kokoa yhdestä alkaen siten, että kaikki luokitin-testiaineisto -yhdistelmät testattiin kaikilla mahdollisilla testiaineistojen kokovaihtoehdoilla. Pienille aineistoille testi ajettiin 100 kertaa ja suurille 10 kertaa. Näin saatiin laskettua testijoukon kokokohtaiset keskimääräiset virheluokitusprosentit ja virheluokitusprosenttien keskihajonnat. Näiden testien tuloksista laadittiin kuvaajat virheluokitusprosentista opetusjoukon koon funktiona. Kuvaajat löytyvät liitteestä A.3. Ensimmäisistä kokoomakuvaajista voidaan päätellä, mikä menetelmä soveltuu parhaiten kyseiseen aineiston luokitteluun milläkin opetusjoukon koolla. Kuvaajista on luettavissa tarkalleen, minkä kokoisilla opetusjoukoilla menetelmien paremmuusjärjestys vaihtuu.

# 6. JATKOTUTKIMUS

Tähän lukuun on kerätty työn aikana esiin tulleita ideoita siitä, millä tavoin luokittimien luokitustarkkuutta voitaisiin parantaa. Lisäksi pohditaan, mitä muutoksia kehitettyyn luokitteluohjelmistoon tulisi tehdä ennen sen liittämistä osaksi konenäkölaadunvalvontajärjestelmää.

Työssä ei ehditty kehittää ja testata luokittelijoille hienostuneita viritysmenetelmiä, sillä tavoite minimoida luokittimien parametrien määrä helppokäyttöisyyden vuoksi oli tärkeämpää. Parempiin luokittelutuloksiin voitaisiin päästä kehittämällä luokittimille parametrien optimointimenetelmä, jonka avulla luokittimien luokittelutarkkuuta voitaisiin parantaa. Caruana ja Niculescu-Mizil [6] listaavat luokitinvertailussaan muutamia tällaisia menetelmiä. Hsu et al. [25] puolestaan esittelevät menetelmän SVM-luokittimien arvojen haarukoimiseen hilan ja ristiinvalidoinnin avulla, joka saattaisi olla yleistettävissä myös muille luokitintyypeille. Samankaltainen itseohjautuvuus voisi myös valita opetusjoukosta osajoukon siten, että valinnalla parannettaisiin yleistämiskykyä ja vältettäisiin luokittimien ylioppiminen. Opetusjoukon alkioiden valikoiminen parantaisi suorituskykyä etenkin pienillä opetusjoukoilla. Duda et al. [16, s. 475-482] esittelevät muutamia tällaisia opetusjoukon optimointia tekeviä menetelmiä, kuten *Adaboost*.

Esitellyssä järjestelmässä luokittelutehtävään sopivat piirreilmaisimet valitaan käsin. Oikean piirreilmaisinyhdistelmän löytäminen voitaisiin aivan hyvin automatisoida, sillä opetusvaiheessa on usein hyvin aikaa suorittaa laskennallisesti raskaitakin operaatioita. Hahmontunnistuksen oppikirjoista [16; 70] löytyy useita menetelmiä käsillä olevaan tehtävään parhaiten soveltuvien piirreilmaisimien valitsemiseksi. Webb [70] on omistanut kirjastaan aiheen käsittelyyn kokonaisen luvun, jossa hän esittelee piirteiden valintaan soveltuvia menetelmiä. Osassa lähestymistavoista luokittimen piirteiden valinta aloitetaan valitsemalla mukaan kaikki käytettävissä olevat piirreilmaisimet, joista sitten karsitaan luokittelun kannalta merkityksettömimmät. Tällöin puhutaan piirreavaruuden dimension alentamisesta (*Reducing dimensionality*). Yleisesti käytettyjä menetelmiä ovat pääkomponenttianalyysi (PCA) [16, s. 568] ja epälineaarinen komponenttianalyysi (NLCA) [16, s. 568], joka on pääkomponenttianalyysiä sopivampi silloin, kun piirteillä on monimutkaisia riippuvuuksia. Dimension alentamis- ja piirteiden valintamenetelmät ovat viime vuosina olleet aktiivisen tutkimuksen kohteena. Aiheeseen kannattaa syventyä esimerkiksi eri menetelmiä ko-

koavan artikkelin avulla. Tällainen on Liun ja Yun [43] artikkeli, jossa vertaillaan erilaisia piirteidenvalintamenetelmiä ja esitellään ehdotus menetelmiä yhdistävän alustan rakentamiseksi. Piirteiden automatisoidun valinnan pitäisi parantaa etenkin naiivin Bayes -luokittimen tarkkuutta, sillä oikein suoritettu piirreavaruuden dimension alentaminen vähentää toisistaan riippuvien piirteiden määrää. Tällöin naiivin Bayes -luokittimen riippumattomuusoletus ei heikennä luokittelun tulosta [22].

Mikäli edellä mainitut tekniikat luokittelutarkkuuden parantamiseksi eivät auta, voidaan luokitteluvirheen minimoimisen sijaan minimoida luokitteluriskiä. Riskin minimoimisen jälkeen suurempi virheluokitusprosentti saattaa olla hyväksyttävä. Luokkakohtaiset päätösfunktiot korvataan tällöin riskifunktioilla ja suurimman päätösfunktion arvon sijaan etsitään pienimmän riskin antava luokitus. Riskiin vaikuttaa virheluokittelun kustannus ja luokittelun varmuus. Duda et al. [16, s. 24-28] esittelevät riskifunktion vain Bayes-luokittimelle, mutta riskifunktion käsite on helposti yleistettävissä myös muille menetelmille. Riskinäkökulman etuna on, että oikeanlainen (pienikustannuksinen) suuri virheluokitusprosentti saattaa olla hyväksyttävissä, jolloin luokittelua voidaan heikosta luokitustarkkuudesta huolimatta käyttää.

Vaikka tyypillisesti laadunvalvonnan luokitteluongelmissa luokkajako tunnetaan luokitinta opetettaessa, tulevana kehityskohteena voisi olla jonkin klusterointimenetelmän testaaminen. Klusterointimenetelmällä saatettaisiin löytää prosessista sellaisia ongelmia (virhetyyppejä), jotka eivät ohjattua oppimista käyttävissä menetelmissä tule ilmi. Hyvänä johdantona aiheeseen toimii Jain et al. [31] kokoama katsaus datan klusterointiin ja siinä käytettyihin menetelmiin.

Tässä työssä ei ole otettu tarkkaa kantaa siihen, miten luokittelijan tulisi toimia osana konenäkölaadunvalvontajärjestelmää. Luokittelijoiden tuottamaa jalostettua virhetietoa voitaisiin näyttää laadunvalvontaohjelmiston operaattoreille näkyvällä näytöllä esimerkiksi Pareto-diagrammissa. Näin tarjottaisiin tuotantoa valvovalle henkilökunnalle välitöntä vastetta tuotannon toiminnasta ja viitteitä mahdollisista tuotantoprosessia vaivaavista ongelmista. Luokittelijan integroiminen osaksi laadunvalvonnan tiedonkeruujärjestelmää mahdollistaa myös raportointityökalujen kehittämisen.

# 7. YHTEENVETO

Tämän työn perusteluna oli tilastollisen luokittelijan konenäkölaadunvalvonnalle tuomat hyödyt. Laadunvalvonta-alan kirjallisuudesta löytyy menestystarinoita siitä, kuinka tuntemalla tyypillisimmät tuotannossa esiintyvät virhetyypit voidaan tuotannon laatua parantaa keskittymällä tuotannon parantamisen kannalta olennaisiin ongelmiin.

Sovelluskohde asetti testatuille luokittelumenetelmille tiettyjä reunaehtoja ja vaatimuksia. Menetelmien tuli olla opetettavissa pienelläkin määrällä virhemalleja, jonka jälkeen niiden tuli pystyä riittävän luotettavasti luokittelemaan konenäköalgoritmeilla löydetyt virheet esimääriteltyihin luokkiin. Menetelmien tuli olla lisäksi riittävän nopeita pysyäkseen tuotannon tahdissa sekä riittävän helppoja ja yksinkertaisia loppuasiakkaan eli tuotannon valvojien käyttää.

Testattavana oli yhteensä neljätoista luokitinvarianttia neljästä eri luokitinperheestä. Approksimoinnilla kiihdytetty lähinaapuriluokitin ja SVM- eli tukivektorikoneluokitin saivat menetelmien välisessä vertailussa parhaat arvosanat. Myös diskretisoitua todennäköisyysjakaumaa käyttävä naiivi Bayes -luokitinvariantti menestyi testeissä hyvin. Muut naiivin Bayes -luokittimen jakaumavaihtoehdot menestyivät vaihtelevasti. Heikoiten luokittimien välisessä vertailussa pärjäsi sumea luokitin, joka kärsi heikosti toteutetusta valmisohjelmistokirjastosta. Sumea luokitin oli menetelmistä myös vaikein testattava, sillä sen oppimismenetelmän kehittämiseen ei työssä ehditty keskittymään riittävästi. Hyvän oppimismenetelmän puuttuessa sumea luokitin vaatisi asiantuntijan virittämään luokittelutehtävään sopivan sääntökannan. Tällaisen koejärjestelyn järjestäminen ei kuitenkaan ollut mahdollista tämän työn aikana. Sumea luokitin on hyödyllinen niissä käytännön tapauksissa, joissa opetusdataa on käytettävissä vain vähän tai ei lainkaan. Käsin opetettavan sumean luokittimen suorituskykypotentiaalia on kuitenkin liki mahdotonta arvioida ilman suurta määrää laadunvalvonnan asiantuntijoita ja heidän avullaan suoritettua kenttätutkimusta.

Edes parhaat oppimis- ja luokittelutarkkuustulokset testeissä saanut SVM-luokitin ei yltänyt asetettujen tavoitteiden mukaiseen virheluokitustarkkuuteen. Luokittelumenetelmien luokittelutarkkuutta voidaan kuitenkin tulevaisuudessa parantaa erilaisin luokitinparametrien, piirteiden ja opetusjoukon valikointimenetelmin. Vaikein työssä käytetty konenäkölaadunvalvonta-alan testiaineisto, jossa oli 112 alkioiset

piirrevektorit ja 16 kohdeluokkaa osoittaa, että osa laadunvalvonnan luokittelutehtävistä on niin vaikeita, ettei täysin luotettavia luokittelupäätöksiä aina voida tehdä. Tällöin voidaan tehdä, kuten jatkokehitysajatuksissa on ehdotettu, eli luokitteluvirheen minimoimisen sijaan minimoidaankin luokitteluriskiä. Riskin minimoimisen jälkeen suurempi virheluokitusprosentti saattaa olla hyväksyttävä.

Luokittimien opetettavuustestit osoittivat, että tyypillisesti konenäkölaadunvalvontaa viritettäessä käytettävissä olevat noin 10 opetusnäytettä on aineiston tilastolliseen analyysiin perustuville menetelmille liian vähän. Edes 50 virhemallia ei riittänyt laadunvalvonta-alan aineistojen tapauksessa tavoitteiden mukaisen virheluokitusprosentin saavuttamiseen. Mikäli konenäkölaadunvalvonnassa halutaan käyttää luokittelua, tulee virhemallien keräämiseen kiinnittää erityistä huomiota. Vain riittävän suuri ja hyvälaatuinen opetusaineisto mahdollistaa tavoitteiden mukaisen virheluokitusprosentin saavuttamisen.

Työn aikana kävi selväksi, että luokittelumenetelmän valinta on aina tehtävä sovelluskohtaisesti. Luokitteluongelman luonteesta ja ominaisuuksista riippuu, millainen luokittelumenetelmä sen ratkaisemiseen soveltuu. Toisissa ongelmissa tarvitaan nopeutta, toisissa tarkkuutta. Työssä toteutettu luokittelijaohjelmisto tarjoaa konenäön ammattilaiselle kattavan joukon erilaisiin luokittelutehtäviin soveltuvia luokittelumenetelmiä.

# LÄHTEET

[1] Arya, S., Mount, D., Netanyahu, N., Silverman, R. & Wu, A. An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions. Journal of the ACM, 45(1998)6, pp. 891-923.

[2] Boser, B.E., Guyon, I.M., & Vapnik, V.N. A training algorithm for optimal margin classifiers. Proceedings of the Fifth Annual Workshop on Computational Learning theory (Pittsburgh, Pennsylvania, United States, July 27 - 29, 1992). COLT '92. ACM, New York, NY, pp. 144-152.

[3] Bouckaert, R.R. Naive Bayes Classifiers That Perform Well with Continuous Variables. Proceedings of the 17th Australian Conference on AI (AI 04), Lecture Notes AI, Berlin 2004. Springer.

[4] Brzankovic, D. & Vujovic, N. Designing a defect classification system: a case study. Pattern Recognition 29(1996)8, pp. 1401-1419.

[5] Carlin, M. Measuring the performance of shape similarity retrieval methods. Computer Vision and Image Understanding 84(2001)1, 44-61.

[6] Caruana, R. & Niculescu-Mizil, A. An Empirical Comparison of Supervised Learning Algorithms. Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, Pennsylvania, USA, 2006. New York, 2006, ACM. pp. 161-168.

[7] Chang, C.-C. & Lin, C.-J. Training $\nu$-Support Vector Classifiers: Theory and Algorithms. Neural Computation 13(2001), pp. 2119-2147.

[8] Chang, C.-C. & Lin, C.-J. LIBSVM: a Library for Support Vector Machines. Taiwan 2009, National Taiwan University. Manual, 30 p.

[9] Chou, P.B., Rao, A.R., Sturzenbecker, M.C., Wu, F.Y. & Brecher, V.H. Automatic defect classification for semiconductor manufacturing. Machine Vision and Applications 9(1997), pp. 201-214

[10] The Code Project Open License (CPOL) v. 1.02 [WWW]. Codeproject 2008. [viitattu 15.2.2010]. Saatavissa: `http://www.codeproject.com/info/cpol10.aspx`.

[11] Cordón, O., del Jesus, M.J. & Herrera, F. A proposal on reasoning methods in fuzzy rule-based classification systems. International Journal of Approximate Reasoning 20(1999), pp. 21-45.

[12] Cortes, C. & Vapnik, V. Support-vector networks. Machine Leaming, 20(1995)3, pp. 273-297.

[13] Cristianini, N. & Shawe-Taylor, J. An Introduction to Support Vector Machines. Cambridge 2000. 189 p.

[14] DeVor, R., Chang, T.-H. & Sutherland, J.W. Statistical Quality Design and Control - Contemporary Concepts and Methods. New York 1992, Macmillan Publishing Company.

[15] Dubois, D. & Prade, H. (eds.), Bezdek, J.C., Keller, J., Krisnapuram, R. & Pal, N.R. Fuzzy Models and Algorithms for Pattern Recognition and Image Processing. New York 2005, Springer. 785 p.

[16] Duda, R., Hart, P. & Stork, D. Pattern Classification, 2nd ed. United States 2001, John Wiley & Sons Inc. 654 p.

[17] Eichhorn, A., Girimonte, D., Klose, A. & Kruse, R. Soft computing for automated suface quality analysis of exterior car body panels. Applied Soft Computing 5(2005), pp. 301-313.

[18] European Machine Vision Association. European Vision Technology Market Statistics 2009, Frankfurt 2009.

[19] Fayyad, U.M. & Irani, K.B. Multi-interval discretization of continuous valued attributes for classification learning. Proceedings of the 13th International Joint Conference on Artificial Intelligence, Chambery (France), 1993. Morgan Kaufmann. pp. 1022-1027.

[20] Fisher, R.A. The Use of Multiple Measurements in Taxonomic Problems. Annals of Eugenics 7 (1936). pp. 179-188.

[21] GNU Lesser General Public License (LGPL) [WWW]. Free Software Foundation 2007. [viitattu 15.2.2010]. Saatavissa: `http://www.gnu.org/copyleft/lesser.html`.

[22] Hand, D.J. & Yu, K. Idiot's Bayes; Not So Stupid after All? International Statistical Review 63(2001)3, pp. 385-398.

[23] Holmström, L., Koistinen, P., Laaksonen, J. & Oja, E. Comparison of Neural and Statistical Classifiers - Theory and Practice. Helsinki 1996, Yliopistopaino, 39 p.

[24] Hsu, C.-W. & Lin, C.-J. A comparison of methods for multi-class support vector machines. IEEE Transactions on Neural Networks, 13(2002)2. pp. 415-425.

[25] Hsu, C.-W., Chang, C-C. & Lin, C.-J. A Practical Guide to Support Vector Classification. Taiwan 2009, National Taiwan University. Guide, 15 p.

[26] Huber, P.J. Robust Statistics. New York 1981, Wiley. 334 p.

[27] Iancu, I. Reasoning system with fuzzy uncertainty. Fuzzy Sets and Systems 92(1997)1, pp. 51-59.

[28] IEC 61131-7, Programmable controllers - Part 7: Fuzzy control programming. Geneva 2000, International Electrotechnical Commission. 114 p.

[29] Iivarinen, J. & Visa, A. An adaptive texture and shape based defect classification. 14th International Conference on Pattern Recognition (ICPR'98), Brisbane, Australia August 16-August 20. icpr, 1(1998), pp. 117-123.

[30] Isomursu, P., Niskanen, V., Carlsson, C. & Eklund, P. Sumean logiikan mahdollisuudet. Helsinki 1993, Tekes julkaisu 34/93. 100 s.

[31] Jain, A.K., Murty, M.N. & Flynn, P.J. Data Clustering : A Review. ACM Computing Surveys 31(1999)3, pp. 264-323.

[32] Jia, H., Murphey, Y.L., Shi, J. & Chang, T.-S. An intelligent real-time vision system for surface defect detection. Proceedings of the 17th International Conference on Pattern Recognition 2004 (ICPR'04) - Volume 3. Washington 2004, IEEE Computer Society. pp. 239-242.

[33] John, G.H. & Langley, P. Estimating continuous distributions in bayesian classifiers. Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence. San Mateo, 1995, Morgan Kaufmann Publishers.

[34] Kailas, K.K., Trinh, B. & Agrawala, A.K. Temporal accuracy and modern high performance processors: A case study using Pentium Pro. Technical Reports from UMIACS, UMIACS-TR-97-60 (1997).

[35] Knerr, S., Personnaz, L., & Dreyfus, G. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In Neurocomputing: Algorithms, Architectures and Applications. Springer-Verlag, 1990.

[36] Koistinen, P. Tilastollinen hahmontunnistus. Helsinki 2002, Helsingin yliopiston matematiikan ja tilastotieteen laitos. Luentomoniste. 130 s.

[37] Kotz, S. & van Dorp, J.R. Beyond Beta. Singapore 2004, World Scientific Publishing Company. 307 p.

[38] Kunttu, I. Shape and gray level descriptors for surface defect image retrieval and classification. Tampere 2002. Tampereen teknillinen Yliopisto. Julkaisu - Tampere University of Technology. Publication 556. 132 p.

[39] Lampinen, J. & Vehtari, A. Bayesilaiset menetelmät hahmontunnistuksessa. In: J. Iivarinen, S. Kaski, ja E. Oja (toim.) Neljännesvuosisata Hatutusta: Hahmontunnistustutkimus Suomessa 1977-2002, Suomen hahmontunnistustutkimuksen seura ry. Otaniemi 2002, Otamedia Oy. s. 86-96.

[40] Lepistö, L. Colour and texture based classification of rock images using classifier combinations. Tampere 2006. Tampereen teknillinen Yliopisto. Julkaisu - Tampere University of Technology. Publication 593. 145 p.

[41] Łeski, J. A Fuzzy If-Then Rule-Based Nonlinear Classifier. International Journal of Applied Mathematics and Computer Science 2(2003)13, pp. 215-223.

[42] Li, Q. & Edwards, J. An enhanced normalisation technique for wavelet shape descriptors. Proceedings of 4$^{\text{th}}$ Conference on Computer and Information Technology, Wihan, China, September 14-16, 2004. pp. 722-729.

[43] Liu, H. & Yu, L. Toward Integrating Feature Selection Algorithms for Classification and Clustering. IEEE Transactions on Knowledge and Data Engineering 17(2005)4, pp. 491-502.

[44] Malamas, E., Petrakis, E., Zervakis, M., Petit, L. & Legat, J.-D. A survey on industrial vision systems, applications and tools. Image and Vision Computing 21(2003)2, pp. 171-188.

[45] Michie, D., Spiegelhalter, D.J. & Taylor, C.C. (eds). Machine Learning, Neural and Statistical Classification [WWW]. 1994, Ellis Horwood [viitattu 22.12.2007]. Saatavissa: http://www.maths.leeds.ac.uk/ charles/statlog/.

[46] The Microsoft Windows User Experience. 1999, Microsoft Press. Microsoft Corporation (Windows User Experience Team). 594 p.

[47] Mital, A., Govindaraj, M. & Subramani, B. A comparison between manual and hybrid methods in parts inspection. Integrated Manufacturing Systems 9(1998)6, pp. 344-349.

[48] Mukherjee, A., Ray, T., Chaudhuri, S., Dutta, P., Sen, S. & Patra, A. Image-based classification of defects in frontal surface of fluted ingot. Measurement 40(2007)6, Pages 687-698.

[49] Newman, T. & Jain, A. A survey of automated visual inspection. Computer vision and image understanding 61(1995)2, pp. 231-262.

[50] Niemi, A. Johdatus sumeisiin joukkoihin ja sumeaan logiikkaan. Helsinki 1996, Opetushallitus. 202 s.

[51] O'Rourke, J. & Toussaint, G. Pattern recognition, Chapter 51. In: Goodman, J. & O'Rourke, J. (eds.) the Handbook of Discrete and Computational Geometry. New York, 2004, Chapman & Hall/CRC, pp. 1135-1162.

[52] Pandelidis, I.O. & Kao, J.F. DETECTOR: A knowledge-based system for injection molding diagnostics. Journal of Intelligent Manufacturing 1(1990), pp. 49-48.

[53] Parzen, E. On Estimation of a Probability Density Function and Mode. The Annals of Mathematical Statistics 33(1962)3, pp. 1065-1076.

[54] Pietikäinen, M., Aikio, H. & Karppinen, K. (eds.). From algorithms to vision systems - Machine vision group 25 years. Oulu 2006, Machine Vision Group. 256 p.

[55] Puolakka, H. Sumea logiikka käytännön sovelluksissa. Helsinki 1997, Opetushallitus. 202 s.

[56] Pöyhönen, S. Support Vector Machine Based Classification in Condition Monitoring of Induction Motors. Dissertation. Helsinki 2004. Helsinki University of Technology. 108 p.

[57] Rabin, S. (ed.). AI Game Programming Wisdom. Hingham MA 2002, Charles River Media Inc. 672 p.

[58] Ripley, B. Neural Networks and Related Methods for Classification. Journal of the Royal Statistical Society. Series B (Methodological), 56(1994)3, pp. 409-456.

[59] Rish, I. An empirical study of the naive Bayes classifier. IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence, pp. 41-16

[60] Ross, T. Fuzzy logic with engineering applications. USA 1995, McGraw-Hill. 600 p.

[61] Roubos, J.A., Setnes, M. & Abonyi, J. Learning fuzzy classification rules from labeled data. Information Sciences 150(2003), pp. 77-93.

[62] Schöolkopf, B., Smola, A., Williamson, R.C. & Bartlett, P.L. New support vector algorithms. Neural Computation, 12(2000). pp. 1207-1245.

[63] Smith, M.L. Surface Inspection Techniques. Suffolk 2001, St. Edmundsbury Press Limited. 198 s.

[64] Tirronen, V., Caponio, A., Haanpää, T. & Meissner, K. Multiple Order Gradient Feature for Macro-Invertebrate Identification Using Support Vector Machines. In: Kolehmainen, M. et al. (Eds.). Adaptive and Natural Computing Algorithms, LNCS 5495. Berlin 2009, Springer-Verlag Berlin Heidelberg. pp. 489-497.

[65] Asuncion, A. & Newman, D.J. UCI Machine Learning Repository [tietokanta]. Irvine 2007. University of California, Irvine, School of Information and Computer Sciences. [viitattu 3.2.2010]. Saatavissa: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[66] Product Quality - A guide for small and medium-sized enterprises. Vienna 2006, United Nations Industrial Development Organization (UNIDO). 56 p.

[67] 3-Clause BSD License. [WWW]. The Regents of the University of California 1999. [viitattu 15.2.2010]. Saatavissa: `http://www.freebsd.org/copyright/license.html`.

[68] Vlachos, M., Domeniconi, C., Gunopulos, D., Kollios, G., & Koudas, N. Nonlinear dimensionality reduction techniques for classification and visualization. Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, Edmonton, Alberta, Canada. New York, NY, 2002. ACM Press. pp. 645-651.

[69] Vuori, V. Hahmontunnistuksen perusteet. Otaniemi 2002. Teknillisen korkeakoulun informaatiotekniikan laitos. Luentomoniste.

[70] Webb, A. Statistical Pattern Recognition (2nd ed.). Chichester 2002, John Wiley & Sons. 495 p.

[71] Wong, B., Elliot, M. & Rapley, C. Automatic casting surface defect recognition and classification. In: IEE Colloquium on Application of Machine Vision. London 1995. pp. 10/1-10/5.

# A. LIITTEITÄ

## A.1 Käytetyt ohjelmistokirjastot

**Windows Platform SDK v. 6.0A** [1] - Windows kehitykseen tarvittavat kirjastot, joita tarvittiin graafisen käyttöliittymän toteuttamiseen. Kirjastot sisältävät myös OpenGL rajapinnan, jolla toteutettiin piirreavaruuden visualisointi. Kirjasto on Microsoftin tekemä ja Windowsin mukana lisensoitu.

**CDataFile v. 2.1** [2] - Ohjelmistokomponentti isojen tekstimuotoisten datatiedostojen lukemiseen ja kirjoittamiseen. Käytetään datajoukon sisäänlukuun. Komponentti on Walter Stormin kehittämä ja lisensoitu The CPOL lisenssillä.

**ANN v. 1.1.1** [3] - *Approximate Nearest Neighbor Library* on kirjasto approksimoidun lähimmän naapurin löytämiseksi. Kirjaston ovat toteuttaneet Sunil Arya ja David Mount. Se on lisensoitu LGPL-lisenssillä.

**LIBSVM v. 2.9** [4] - Chih-Chung Changin ja Chih-Jen Linin kehittämä kirjasto, joka toteuttaa sekä ykköstyypin ($C$-SVM), että kakkostyypin ($\nu$-SMV) SMV luokittimet. Kirjastoa jaellaan LGPL-lisenssillä.

**FFLL v. 2.2.1** [5] - *The Free Fuzzy Logic Library* on kirjasto, joka toteuttaa sumean päättelykoneen. Se tukee IEC 61131-7 [28] standardin mukaisia sumean ohjauskielen FCL tiedostoja. Kirjasto on lisensoitu BSD -lisenssillä.

**windev v. 1.0** [6] - *The Personal Framework for Object Oriented Windows Development* on Stefano Grassin Microsoft Windows -ohjelmistojen rakentamista helpottava kirjasto, joka tarjoaa rajapinnat käyttöliittymäkomponenttien ohjelmointiin. Kirjasto on täysin vapaasti lisensoitu, eli sen käytössä ei ole mitään rajoituksia.

**Ohjelmistokirjastojen viiteluettelo on seuraavalla sivulla.**

1. WindowsSDK Microsoft Windows SDK 6.0A [WWW]. Redmond 2006. [viitattu 15.2.2010]. Saatavissa: http://msdn.microsoft.com/fi-fi/windows/bb980924%28en-us%29.aspx.

2. CDatafile Storm W. CDataFile v. 2.1 - An easy class for reading numeric data in CSV or Text-Delimited format [WWW]. Codeproject 2004. [viitattu 15.2.2010]. Saatavissa: http://www.codeproject.com/KB/files/cdatafile.aspx.

3. ANN Mount D.M. & Arya S. Approximate Nearest Neighbor Library v. 1.1.1 [WWW]. University of Maryland 2009. [viitattu 15.2.2010]. Saatavissa: http://www.cs.umd.edu/~mount/ANN/.

4. LIBSVM Chang C.-C. & Lin C.-J. LIBSVM - A Library for Support Vector Machines v. 2.9 [WWW]. Taiwan 2009, National Taiwan University. [viitattu 15.2.2010]. Saatavissa: http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

5. FFLL The Free Fuzzy Logic Library v. 2.2.1 [WWW]. Louder Than A Bomb! Software 2003. [viitattu 15.2.2010]. Saatavissa: http://ffll.sourceforge.net/.

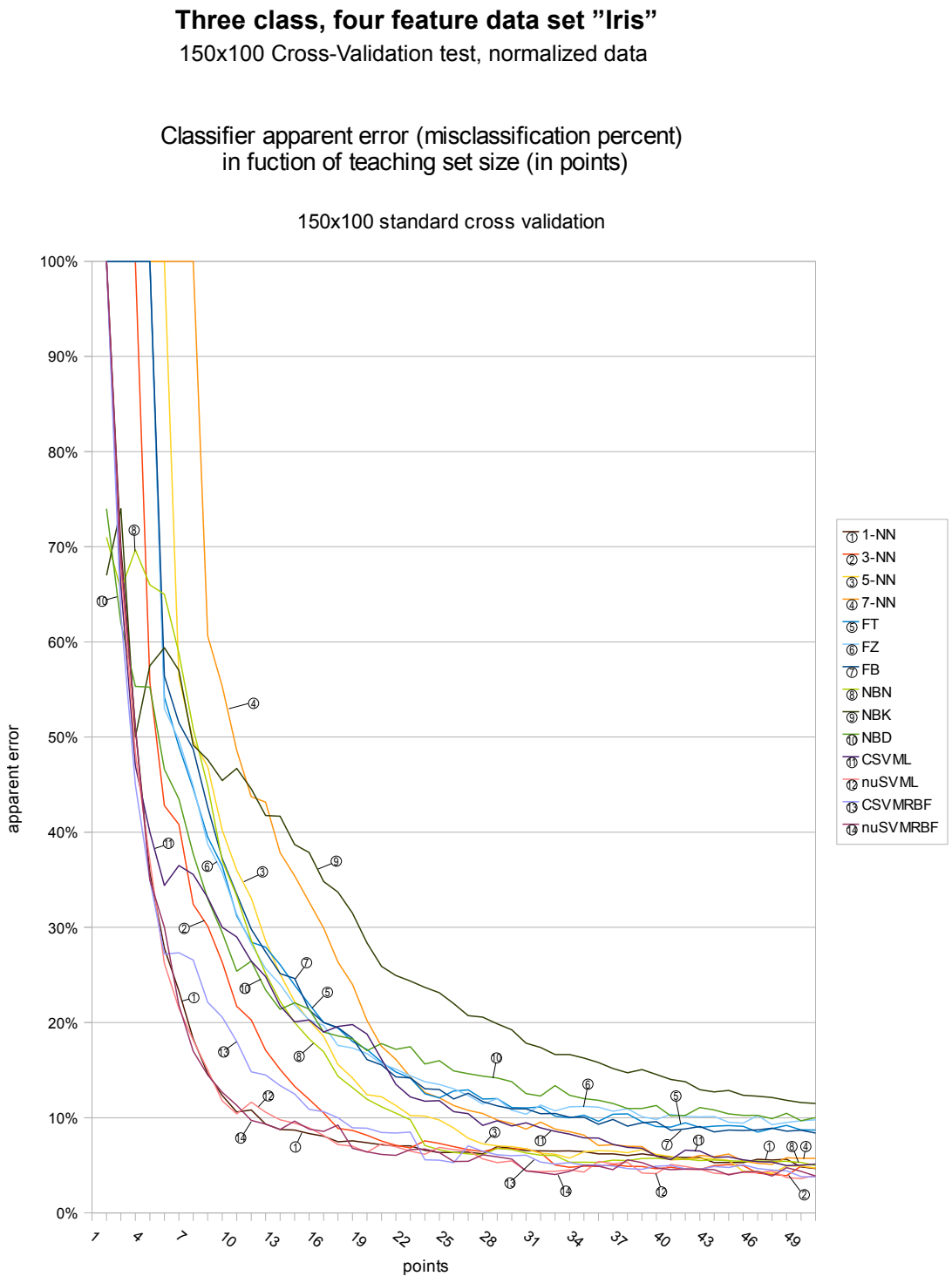
6. WinDev Grassi S. The Personal Framework for Object Oriented Windows Development v. 1.0 [WWW]. SGR 2003. [viitattu 15.2.2010]. Saatavissa: http://www.sgr.info/dev/windev/main.htm.

## A.2 Esimerkkilähdekoodi luokittelukirjaston käytöstä

C:\MyTemp\juherask\projects\transopt\publications\juherask\trunk\docs\usage.cpp 8. tammikuuta 2010 17:35

```
/**
 * prDataSet is a pointer to a full dataset loaded from file.
 * clfID is an 0 based index of an classifier to use.
 * Fuction returns pointer to half of the prDataSet classified using
 *  an classifier trained with other half of the dataset.
 **/
CFeatureVectorSet* ClassifierUsageExample
(
    CFeatureVectorSet* prDataSet,
    int clfID
)
{
    CFeatureVectorSet* pTrainSet = NULL;
    CFeatureVectorSet* prTestSet = NULL;
    IClassifier* pClassifier = NULL;
    int vc = prDataSet->GetVectorCount();

    // Create new classifier
    pClassifier = CClassifierFactory::BuildNewClassifier(clfID);

    // pTrainSet has 50% randomly selected samples from data set.
    pTrainSet = new CFeatureVectorSet( *prDataSet, (int)(vc*0.50) );
    // pTestSet has those samples not in pTrainSet.
    prTestSet = new CFeatureVectorSet(*prDataSet, *pTrainSet);

    // Train & Classify
    try
    {
        pClassifier->Train( pTrainSet );
        pClassifier->Classify( prTestSet );
    }
    catch (CClassifierException* ex)
    {
        cerr << ex->GetMessage();
        delete prTestSet; prTestSet = NULL;
    }

    // Free memory
    delete pClassifier;
    delete pTrainSet;

    // prTestSet has now been classified using prClassifier
    return prTestSet;
}
```

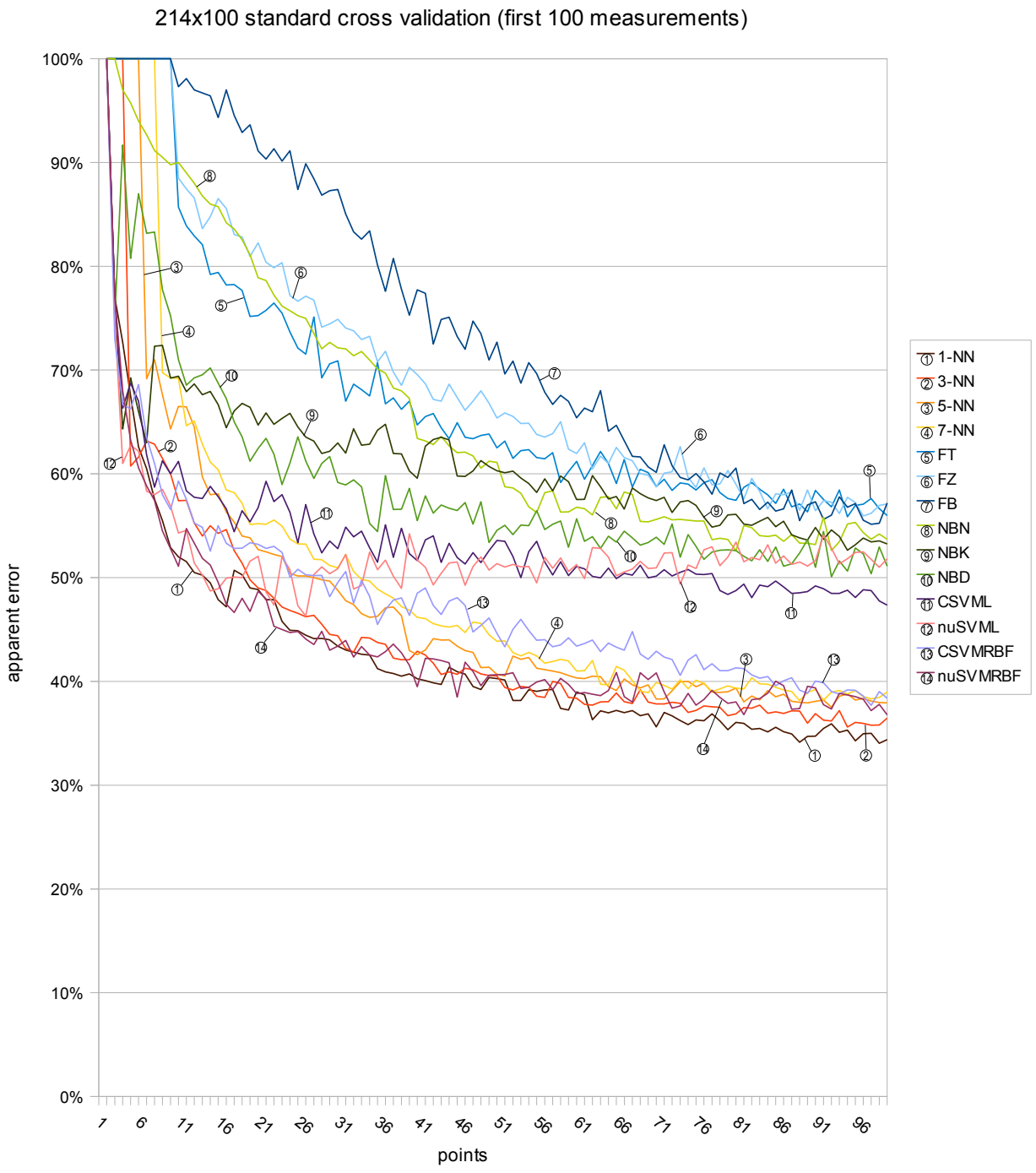
## A.3 Testiaineiston täyden luokittelun kuvaajat



## Seven class, nine feature data set ”glass”

214x100 Cross-Validation test, normalized data

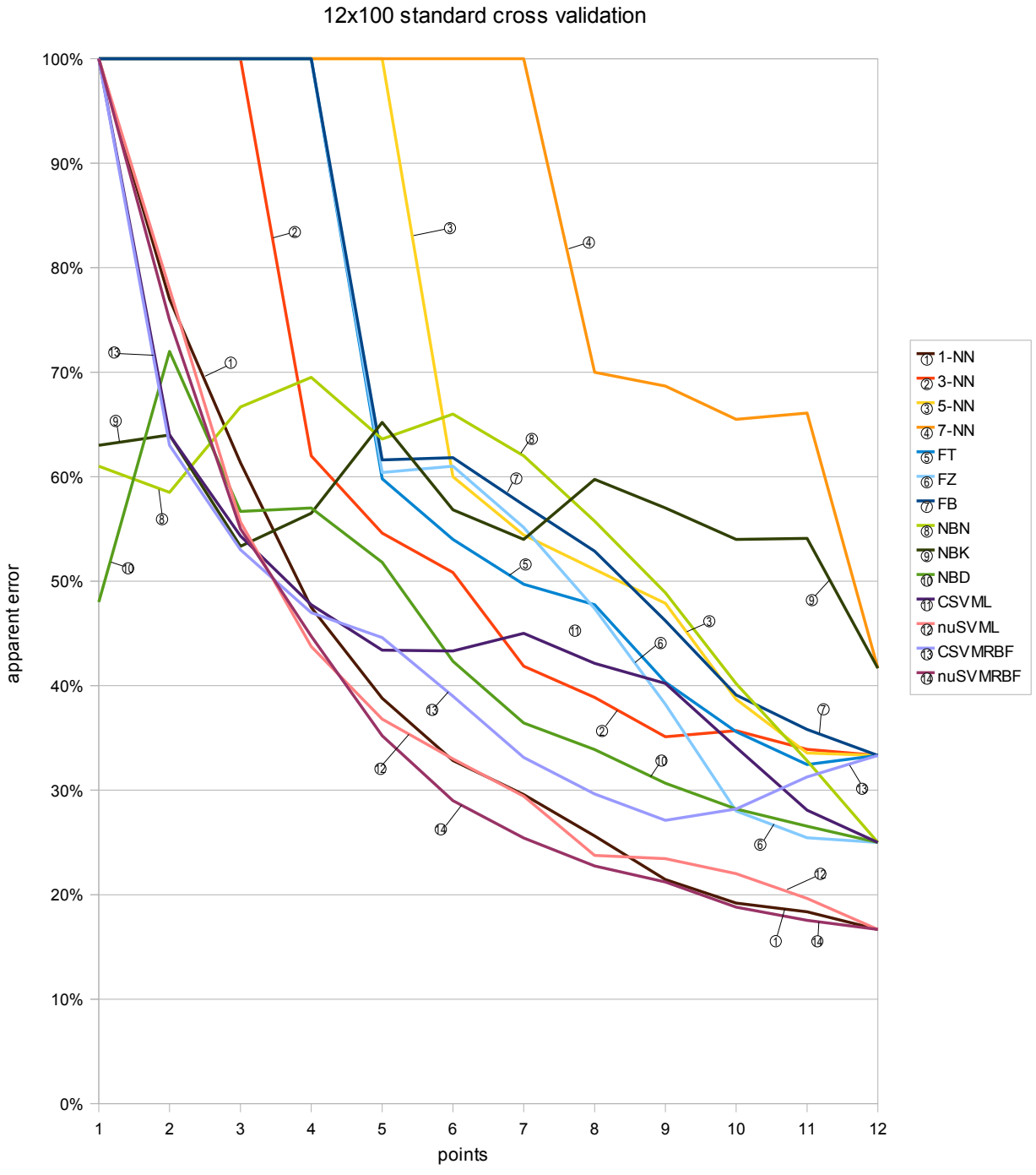
Classifier apparent error (misclassification percent)
in fuction of teaching set size (in points)



# Three class, four feature data set ”Casting”

12x100 Cross-Validation test, normalized data

Classifier apparent error (misclassification percent)
in fuction of teaching set size (in points)

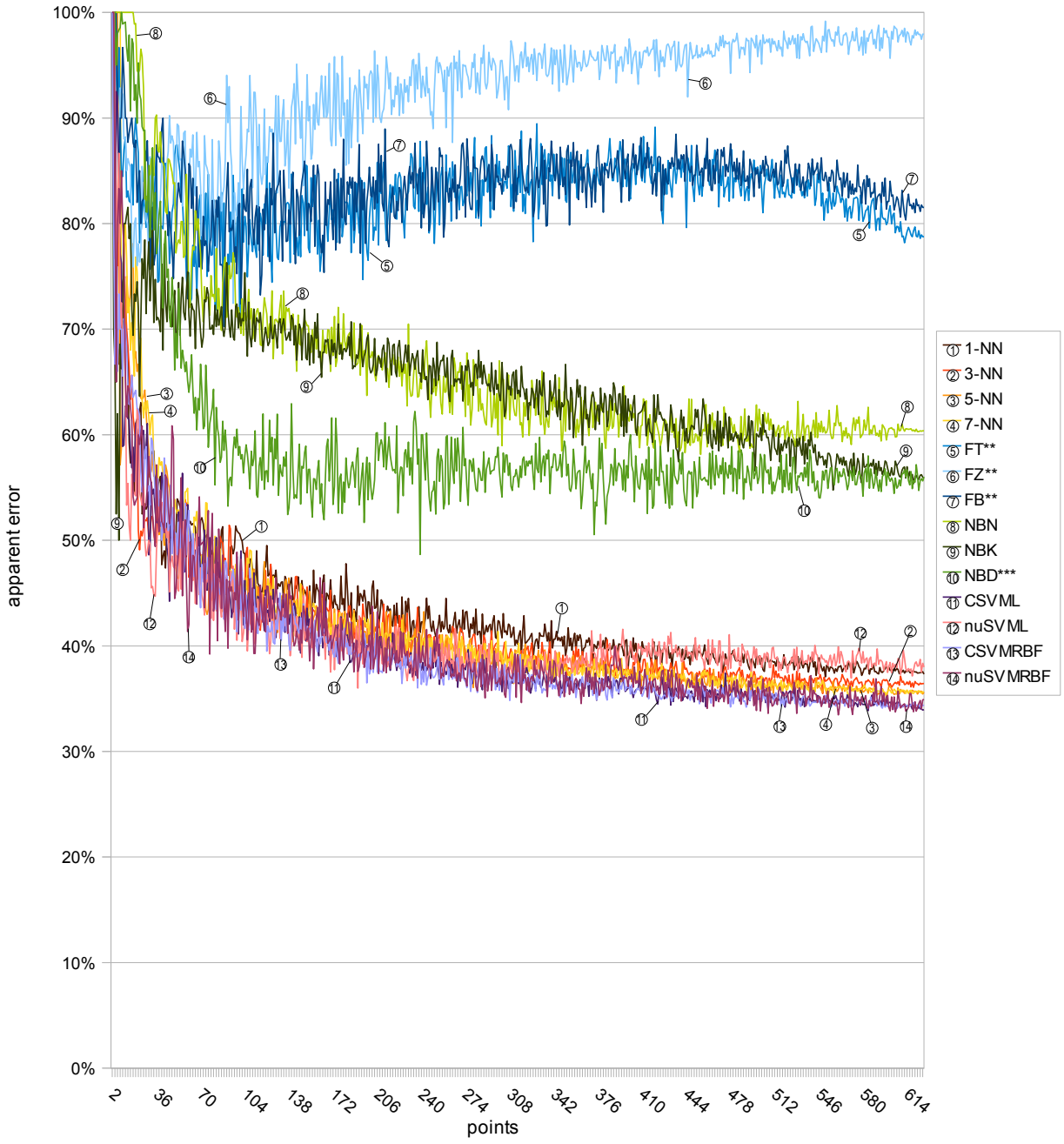


# 16 class, 112 feature data set ”Paper”

626x10 Cross-Validation test, normalized data

Classifier apparent error (misclassification percent)
in fuction of teaching set size (in points)

626x10 standard cross validation
** are results with trimmed feature vector
*** are 5 x 10-fold cross validation

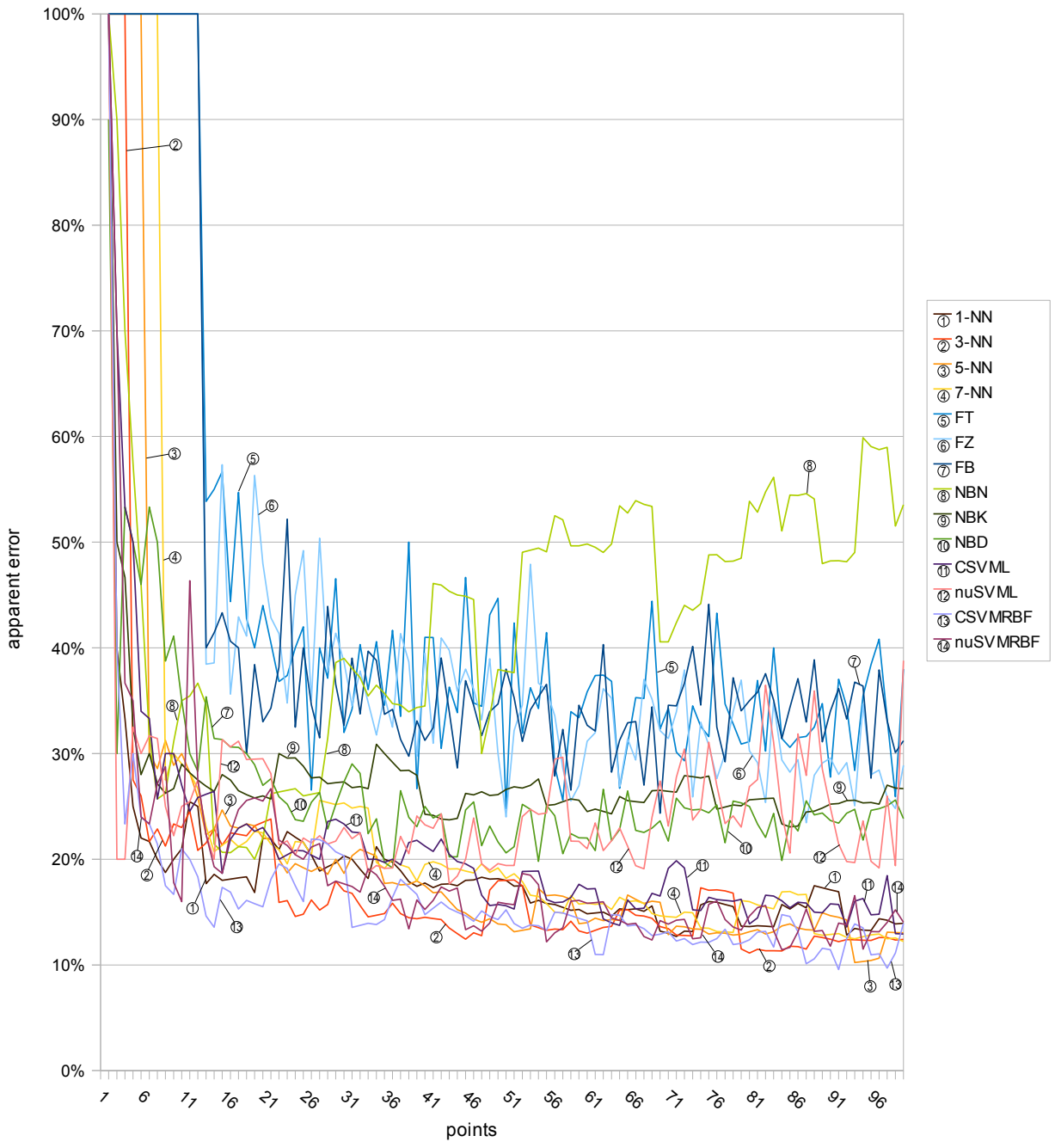


## 5 class, 12 feature data set ”IMD”

213x100 Cross-Validation test, normalized data

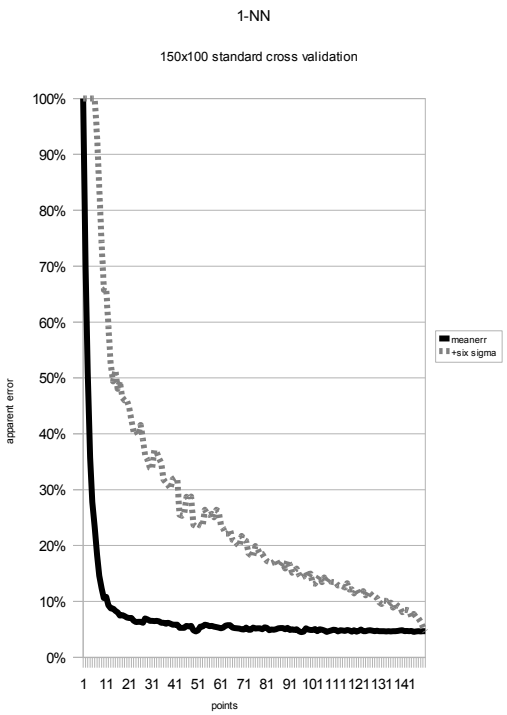
Classifier apparent error (misclassification percent)
in fuction of teaching set size (in points)

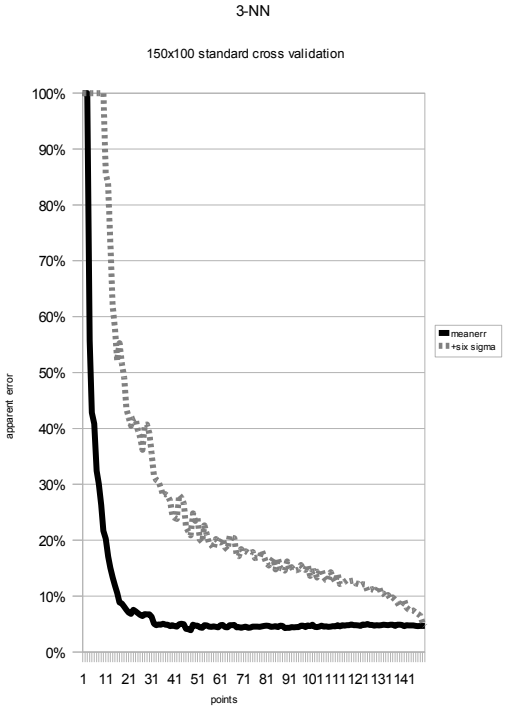
213x100 standard cross validation (first 100 measurements)

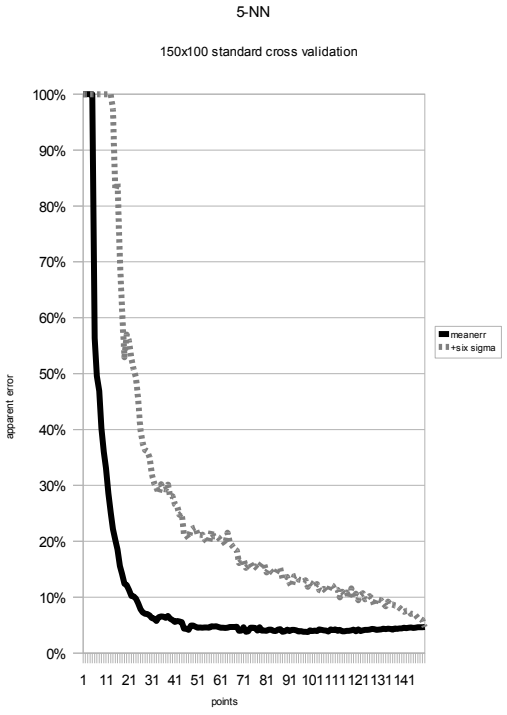


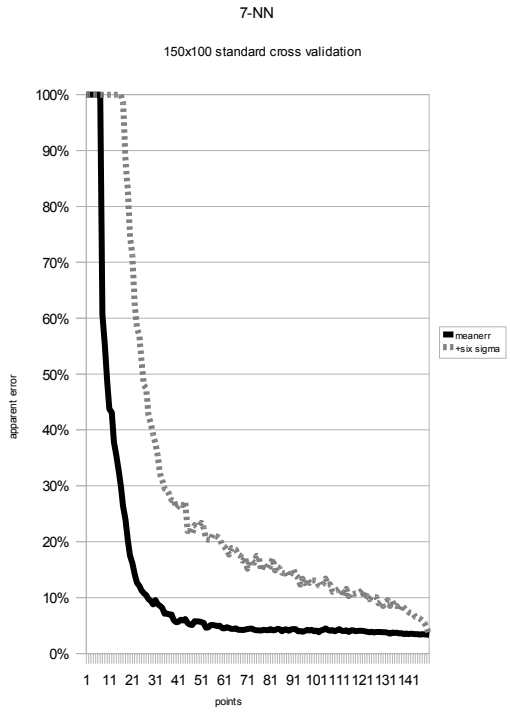
Iris

# K-NN

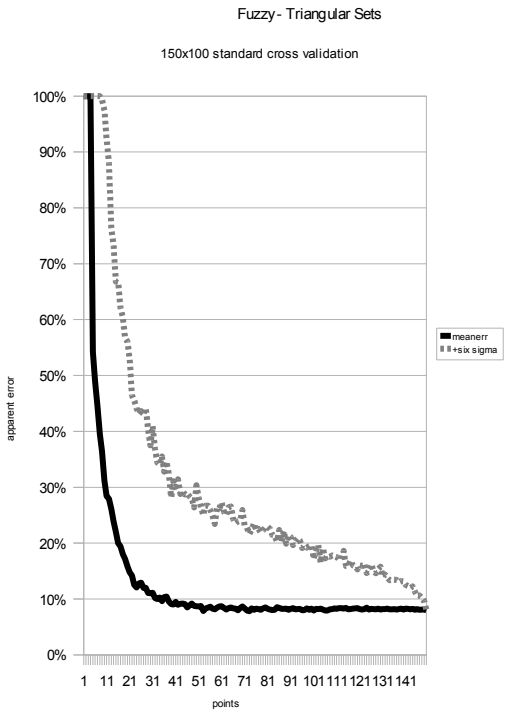

1-NN
150x100 standard cross validation
apparent error
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 11 21 31 41 51 61 71 81 91 101 111 121 131 141
points
meanerr
+six sigma

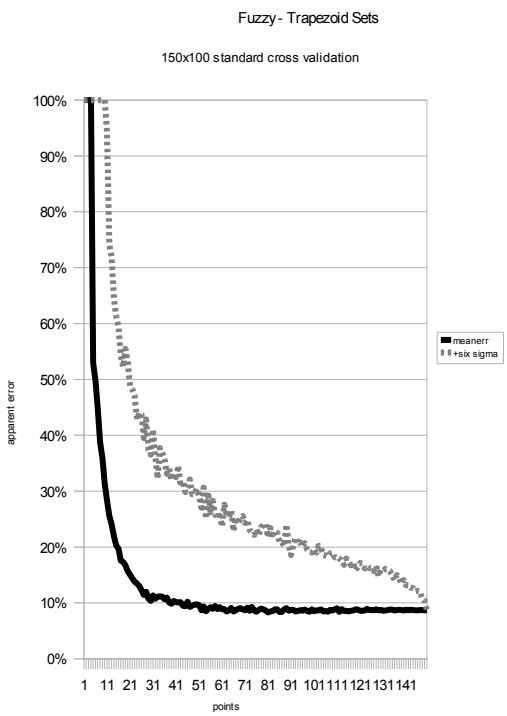

3-NN
150x100 standard cross validation
apparent error
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 11 21 31 41 51 61 71 81 91 101 111 121 131 141
points
meanerr
+six sigma

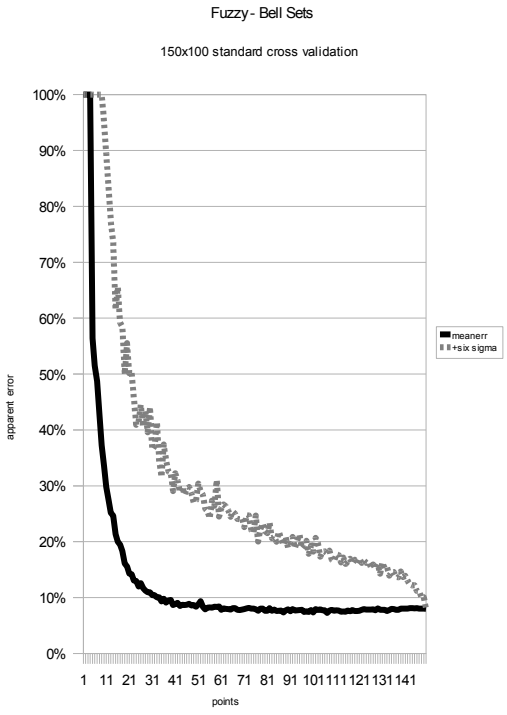

5-NN
150x100 standard cross validation
apparent error
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 11 21 31 41 51 61 71 81 91 101 111 121 131 141
points
meanerr
+six sigma


7-NN
150x100 standard cross validation
apparent error
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 11 21 31 41 51 61 71 81 91 101 111 121 131 141
points
meanerr
+six sigma

# Fuzzy

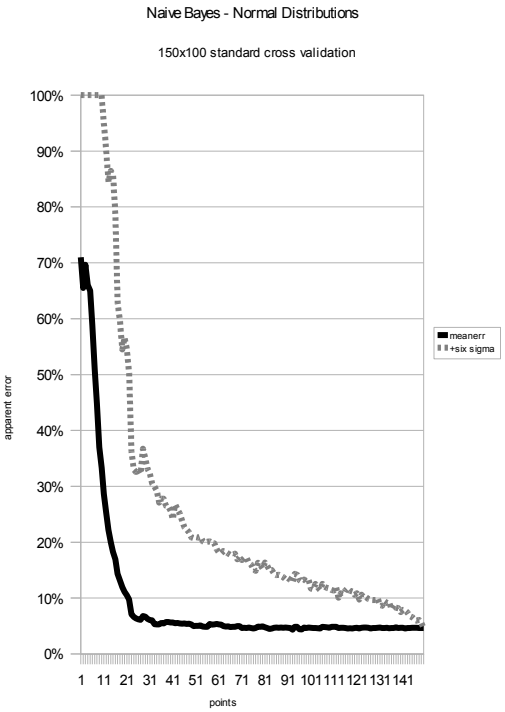

Fuzzy - Triangular Sets
150x100 standard cross validation
apparent error
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 11 21 31 41 51 61 71 81 91 101 111 121 131 141
points
meanerr
+six sigma

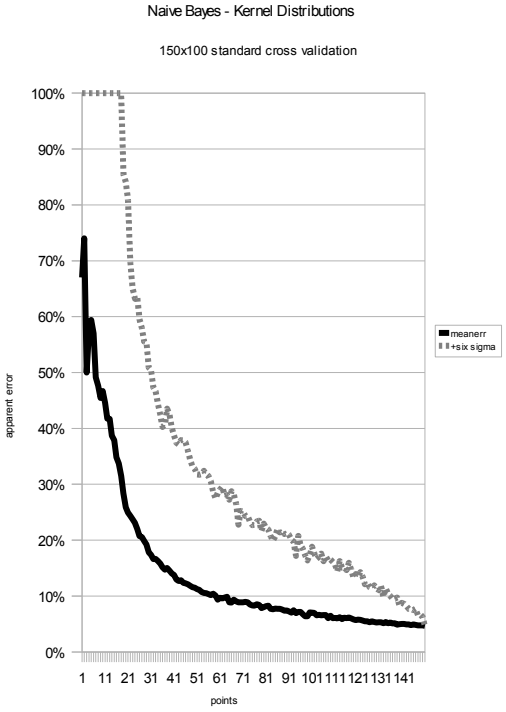

Fuzzy - Trapezoid Sets
150x100 standard cross validation
apparent error
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 11 21 31 41 51 61 71 81 91 101 111 121 131 141
points
meanerr
+six sigma

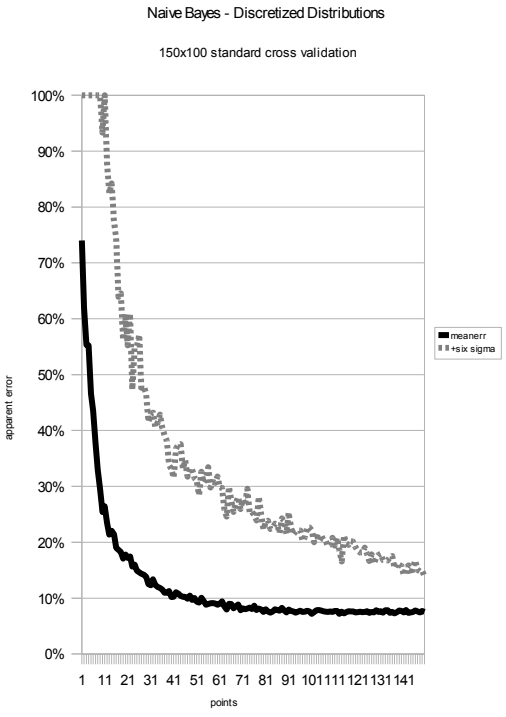

Fuzzy - Bell Sets
150x100 standard cross validation
apparent error
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 11 21 31 41 51 61 71 81 91 101 111 121 131 141
points
meanerr
+six sigma

Iris

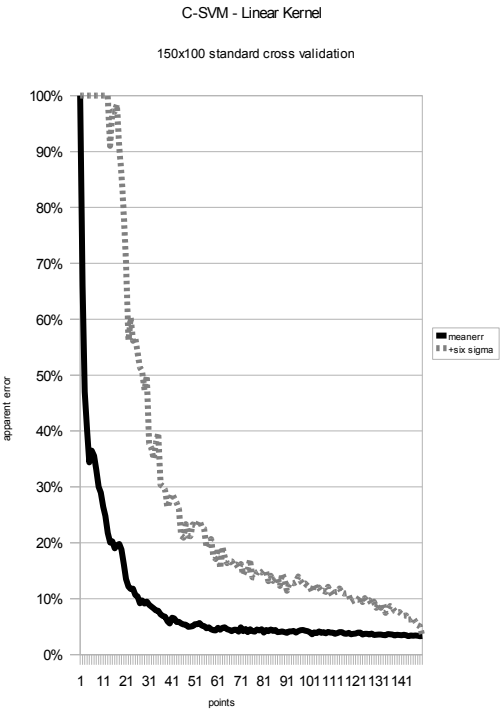
# Naive Bayes

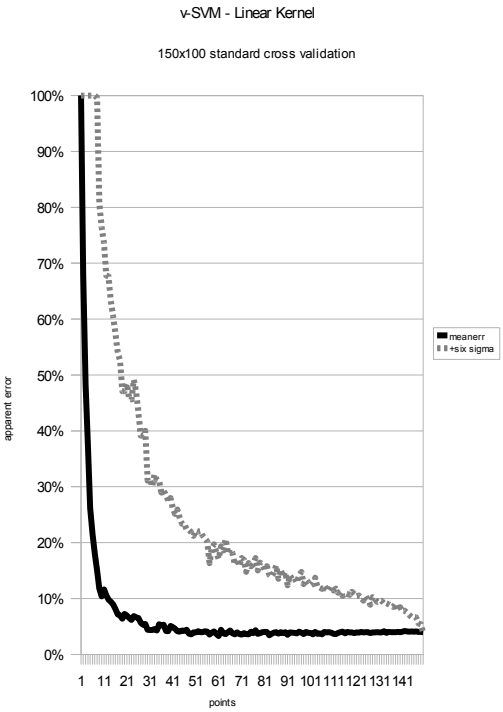


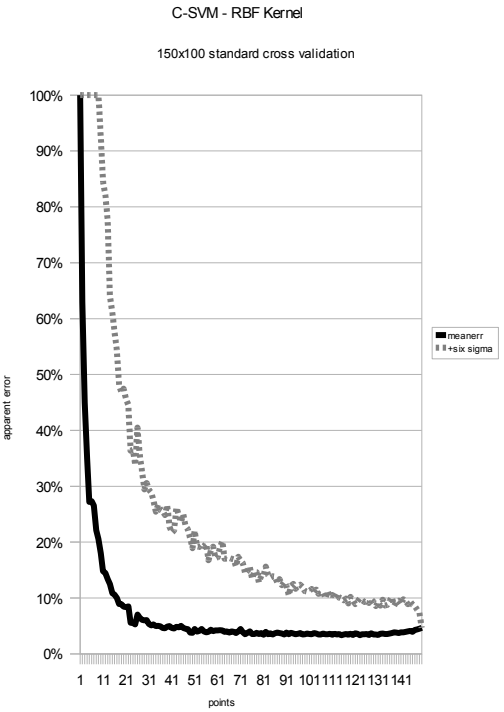



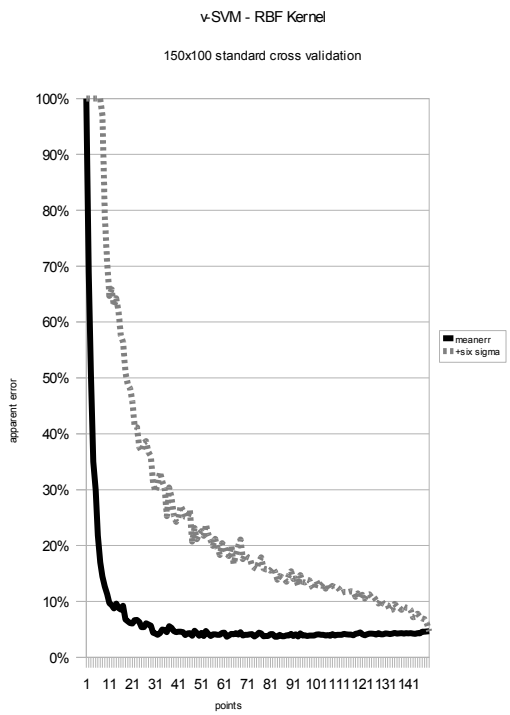



# SVM


C-SVM - Linear Kernel
150x100 standard cross validation
apparent error
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 11 21 31 41 51 61 71 81 91 101 111 121 131 141
points
meanerr
+six sigma

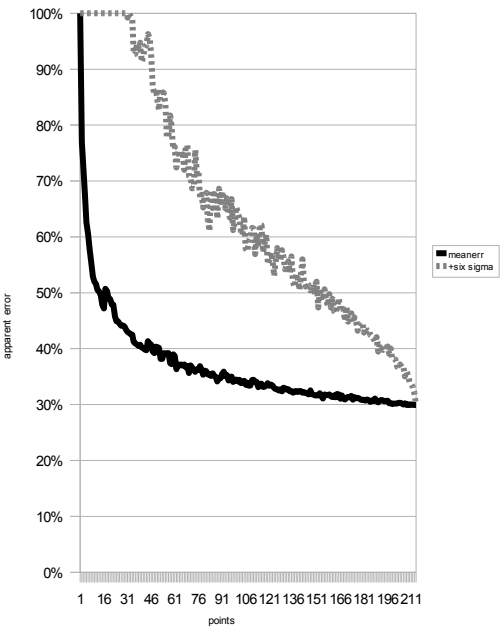

v-SVM - Linear Kernel
150x100 standard cross validation
apparent error
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 11 21 31 41 51 61 71 81 91 101 111 121 131 141
points
meanerr
+six sigma


C-SVM - RBF Kernel
150x100 standard cross validation
apparent error
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 11 21 31 41 51 61 71 81 91 101 111 121 131 141
points
meanerr
+six sigma

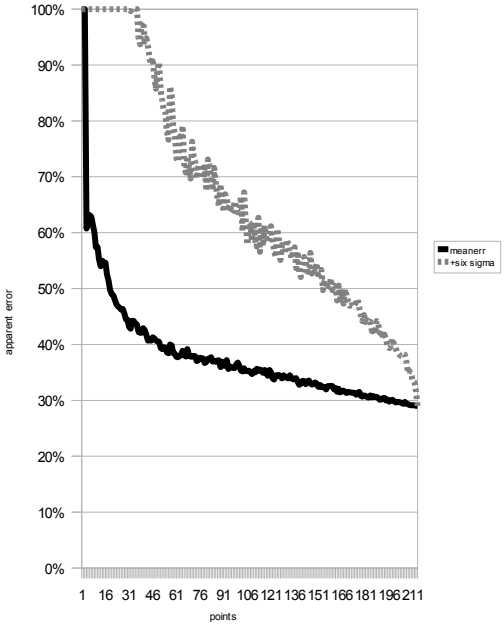

v-SVM - RBF Kernel
150x100 standard cross validation
apparent error
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 11 21 31 41 51 61 71 81 91 101 111 121 131 141
points
meanerr
+six sigma

# K-NN

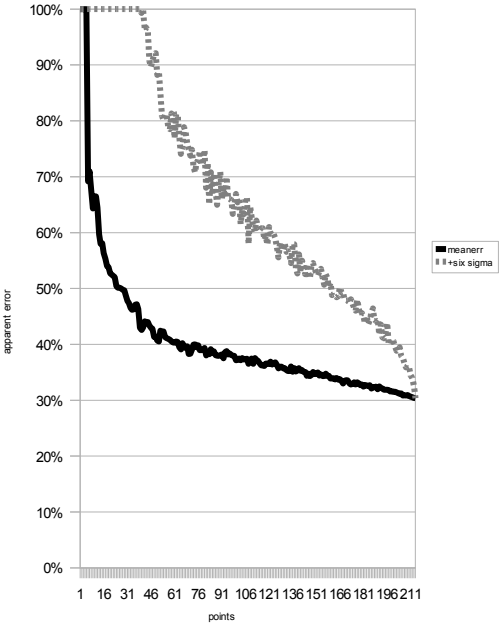
1-NN



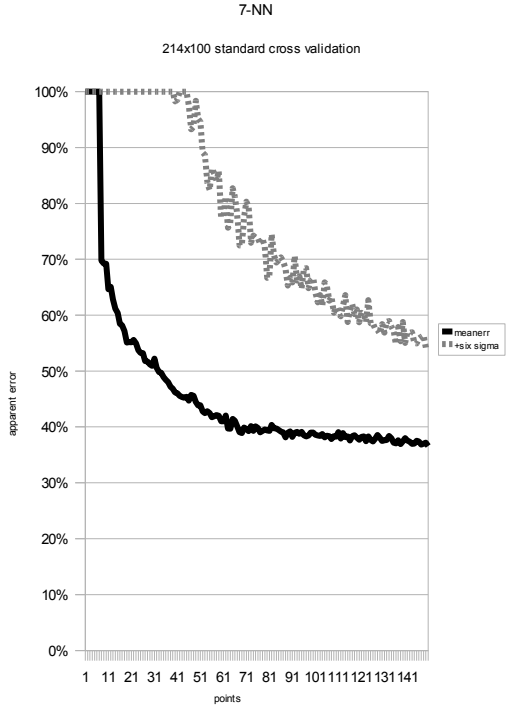

3-NN



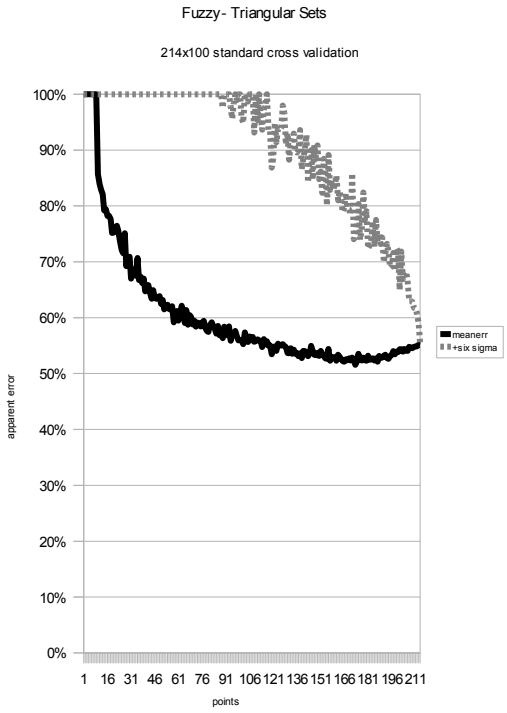

5-NN

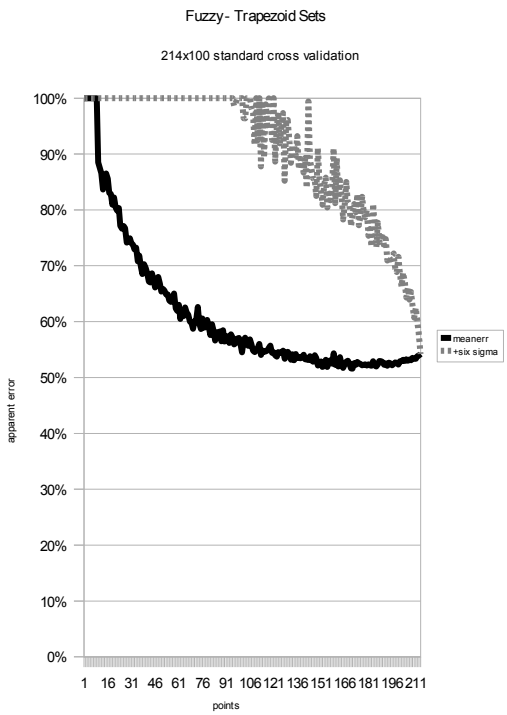


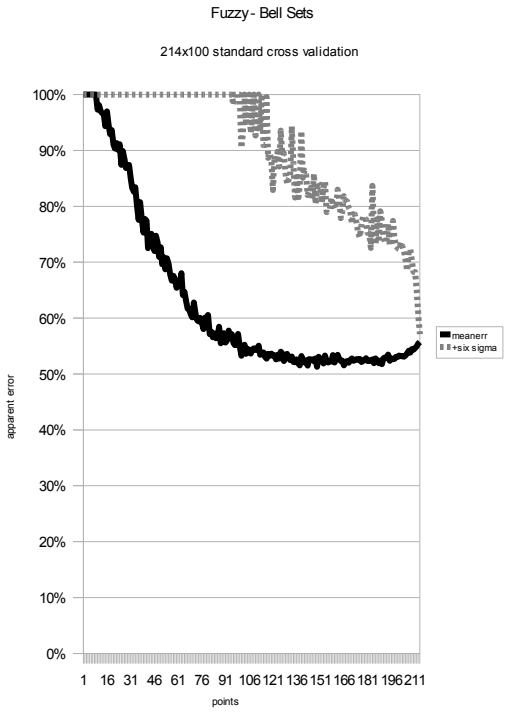

7-NN



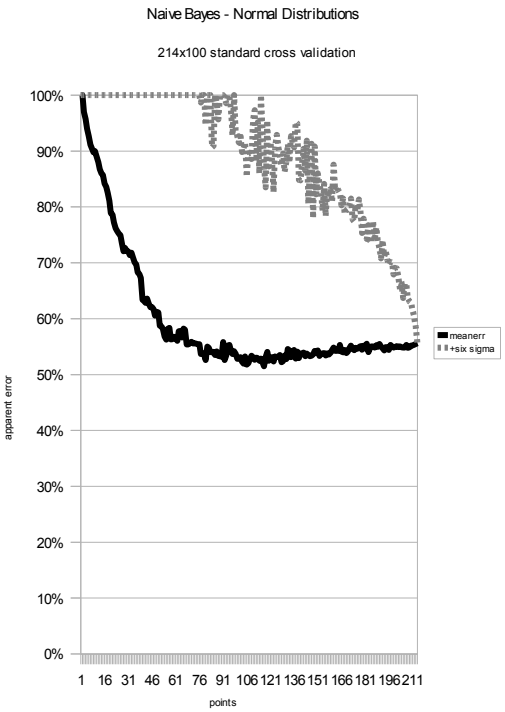
# Fuzzy

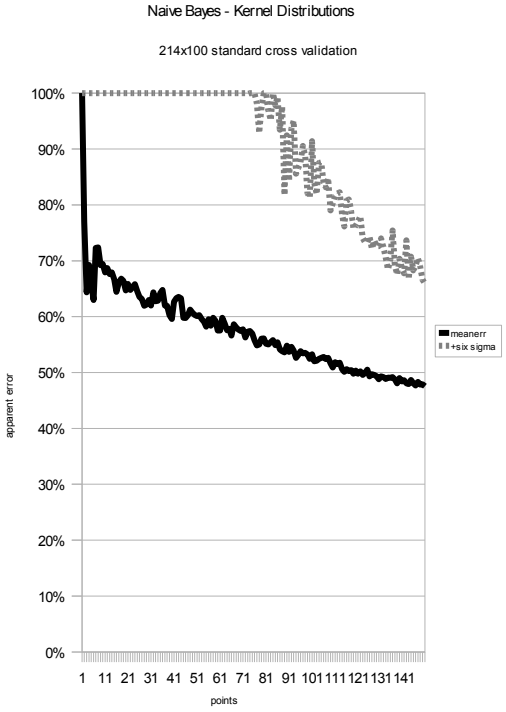

Fuzzy- Triangular Sets
214x100 standard cross validation
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
apparent error
1 16 31 46 61 76 91 106 121 136 151 166 181 196 211
points
meanerr
+six sigma

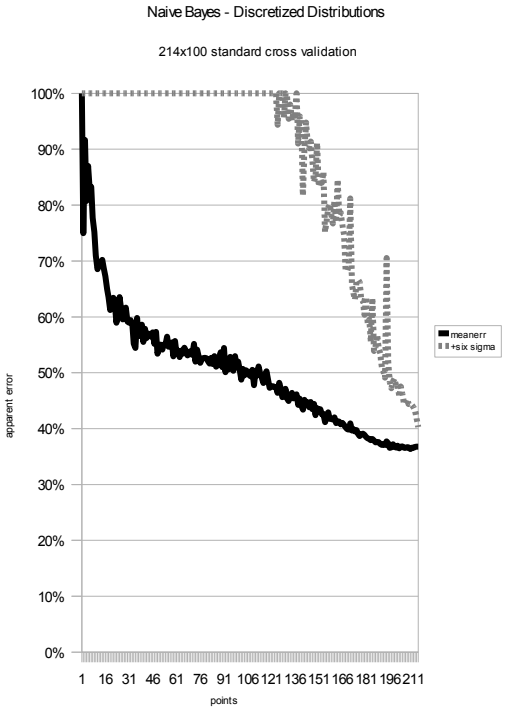

Fuzzy- Trapezoid Sets
214x100 standard cross validation
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
apparent error
1 16 31 46 61 76 91 106 121 136 151 166 181 196 211
points
meanerr
+six sigma


Fuzzy- Bell Sets
214x100 standard cross validation
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
apparent error
1 16 31 46 61 76 91 106 121 136 151 166 181 196 211
points
meanerr
+six sigma

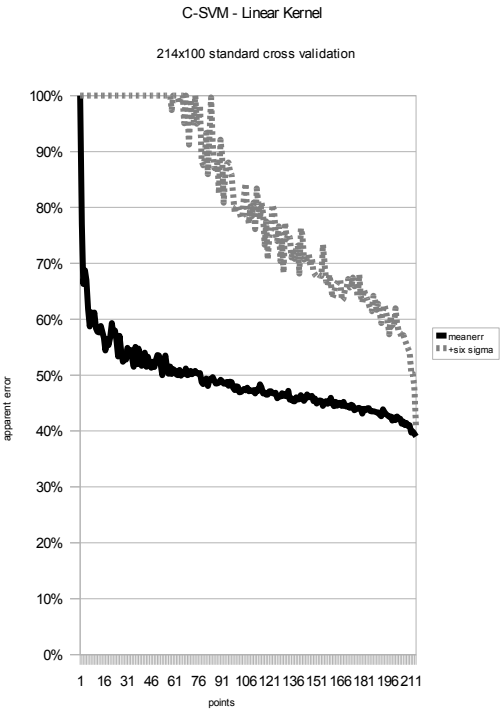
# Naive Bayes

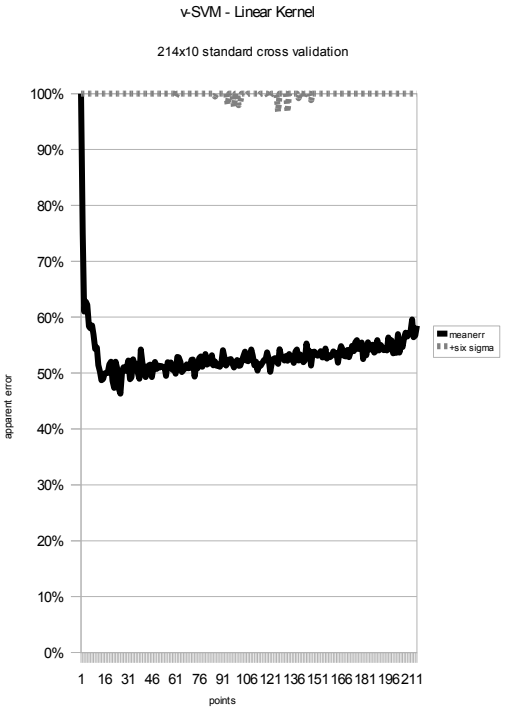

Naive Bayes - Normal Distributions
214x100 standard cross validation
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
apparent error
1 16 31 46 61 76 91 106 121 136 151 166 181 196 211
points
meanerr
+six sigma

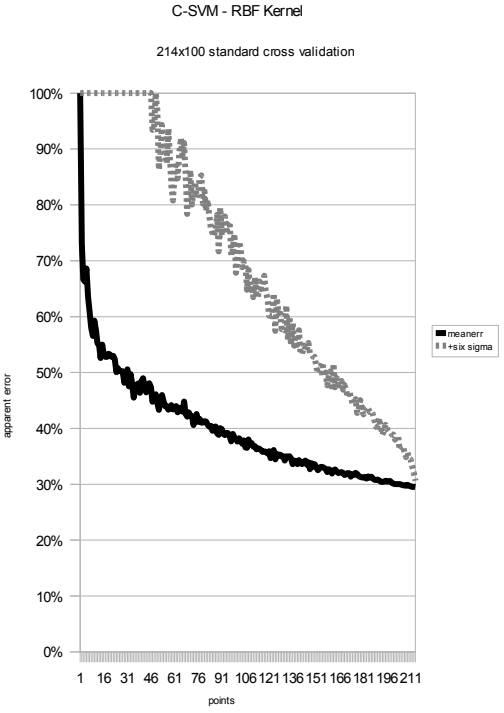

Naive Bayes - Kernel Distributions
214x100 standard cross validation
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
apparent error
1 11 21 31 41 51 61 71 81 91 101 111 121 131 141
points
meanerr
+six sigma

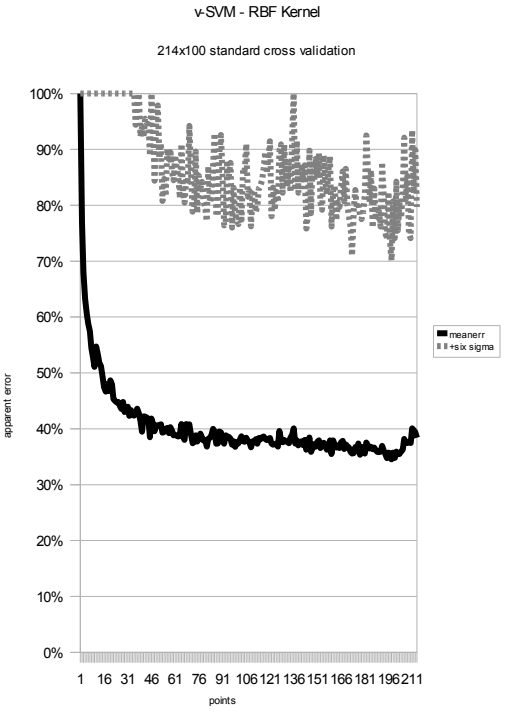

Naive Bayes - Discretized Distributions
214x100 standard cross validation
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
apparent error
1 16 31 46 61 76 91 106 121 136 151 166 181 196 211
points
meanerr
+six sigma

# SVM


C-SVM - Linear Kernel
214x100 standard cross validation
apparent error
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 16 31 46 61 76 91 106 121 136 151 166 181 196 211
points
meanerr
+six sigma

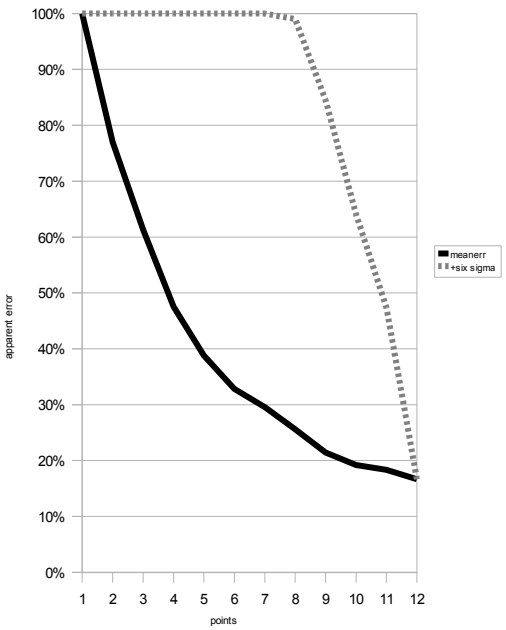

v-SVM - Linear Kernel
214x10 standard cross validation
apparent error
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 16 31 46 61 76 91 106 121 136 151 166 181 196 211
points
meanerr
+six sigma


C-SVM - RBF Kernel
214x100 standard cross validation
apparent error
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 16 31 46 61 76 91 106 121 136 151 166 181 196 211
points
meanerr
+six sigma

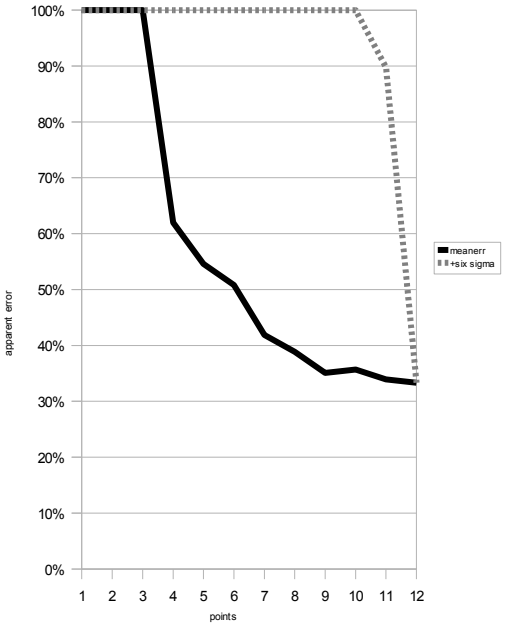

v-SVM - RBF Kernel
214x100 standard cross validation
apparent error
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 16 31 46 61 76 91 106 121 136 151 166 181 196 211
points
meanerr
+six sigma

# K-NN

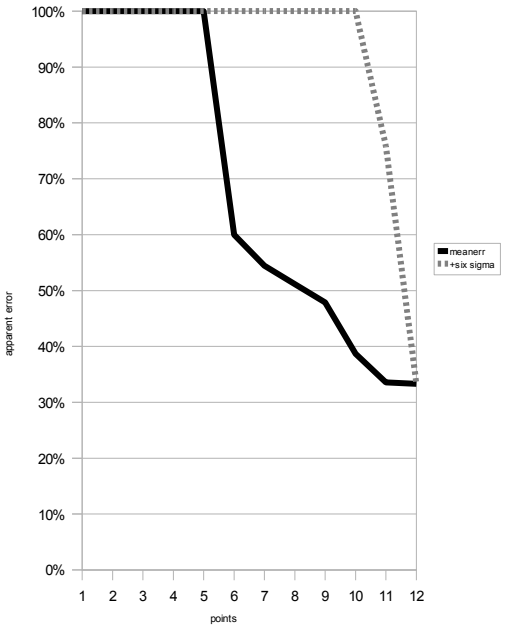
1-NN



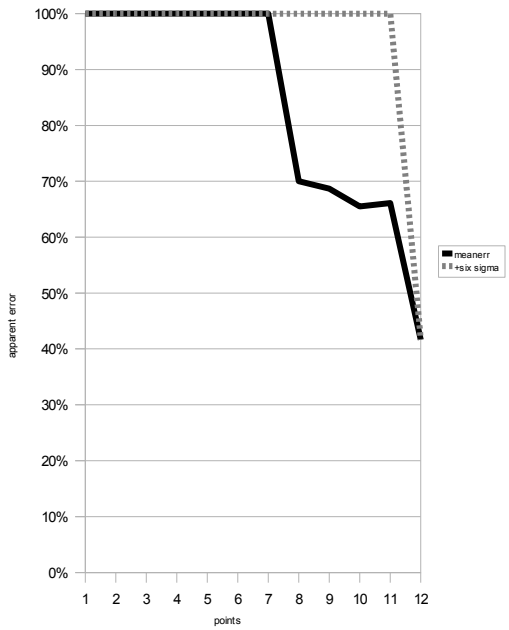

3-NN



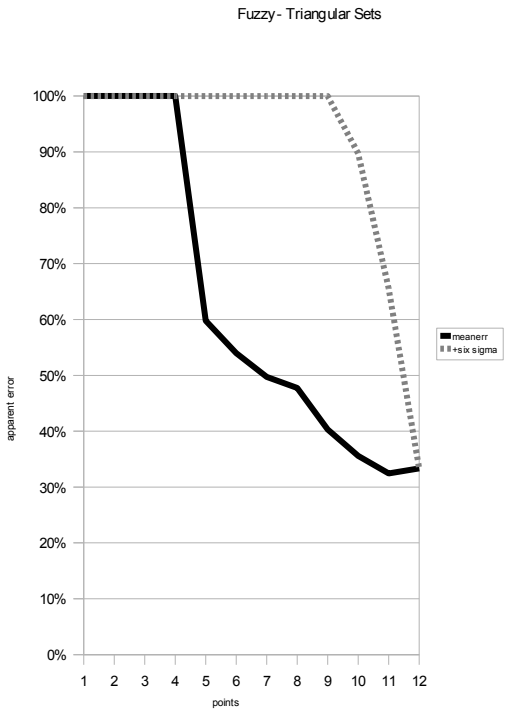

5-NN

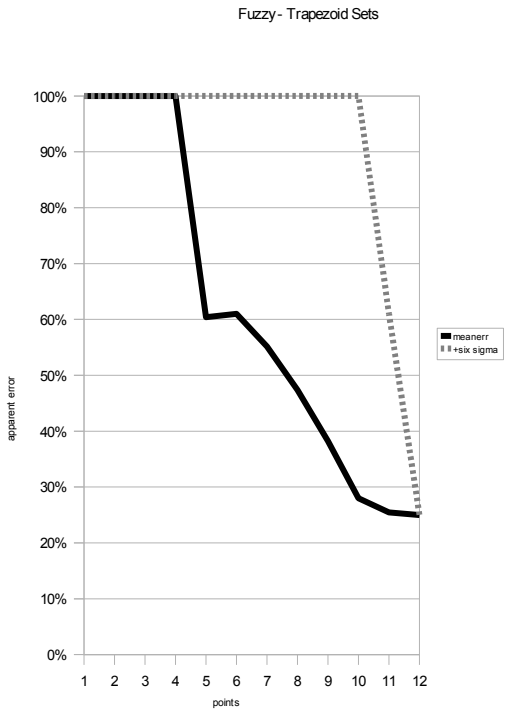


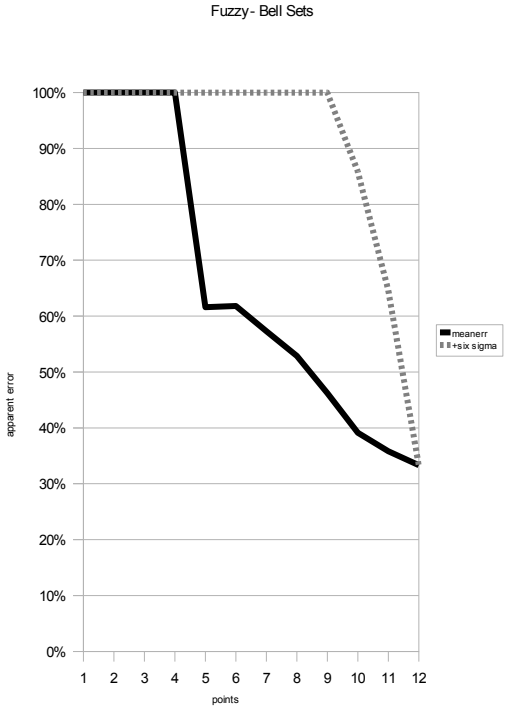

7-NN



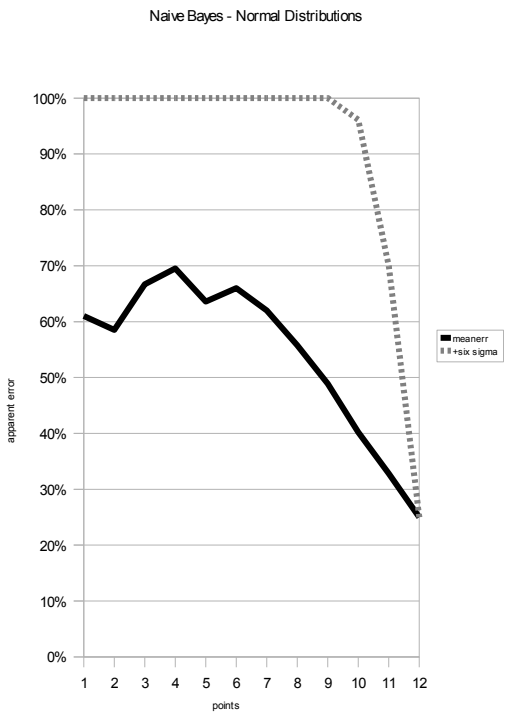
# Fuzzy

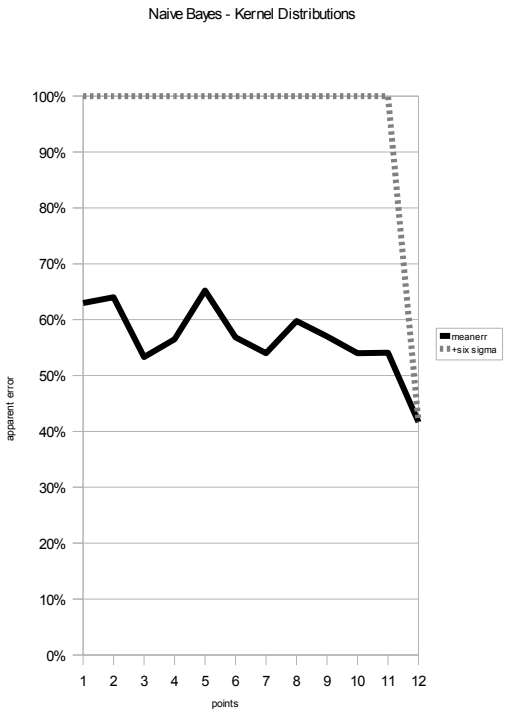

Fuzzy - Triangular Sets
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
apparent error
1 2 3 4 5 6 7 8 9 10 11 12
points
meanerr
+six sigma

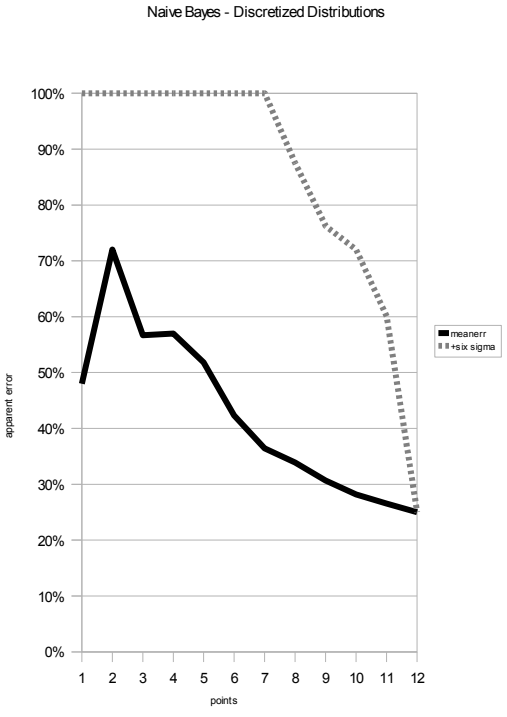

Fuzzy - Trapezoid Sets
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
apparent error
1 2 3 4 5 6 7 8 9 10 11 12
points
meanerr
+six sigma


Fuzzy - Bell Sets
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
apparent error
1 2 3 4 5 6 7 8 9 10 11 12
points
meanerr
+six sigma

# Naive Bayes

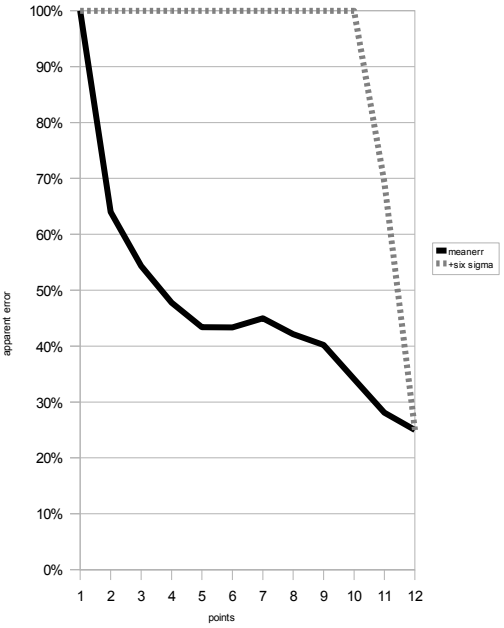

Naive Bayes - Normal Distributions
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
apparent error
1 2 3 4 5 6 7 8 9 10 11 12
points
meanerr
+six sigma


Naive Bayes - Kernel Distributions
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
apparent error
1 2 3 4 5 6 7 8 9 10 11 12
points
meanerr
+six sigma

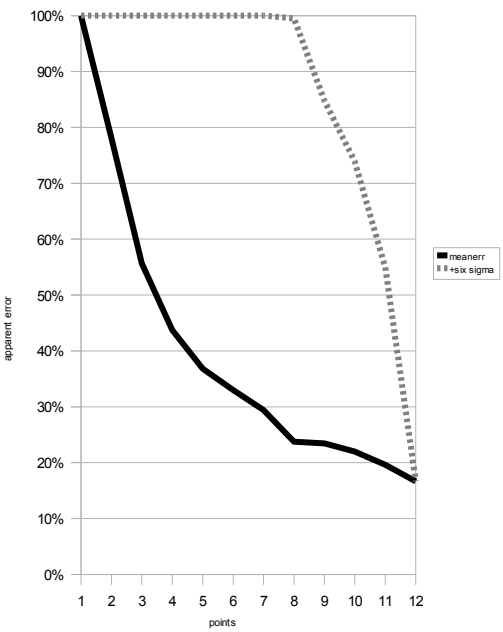

Naive Bayes - Discretized Distributions
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
apparent error
1 2 3 4 5 6 7 8 9 10 11 12
points
meanerr
+six sigma

# SVM

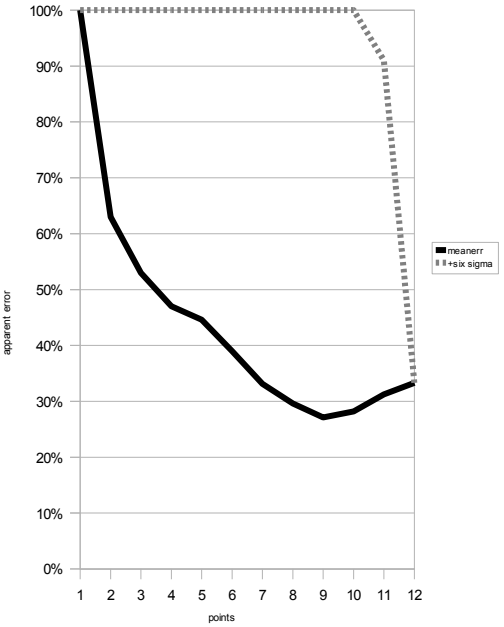
C-SVM - Linear Kernel



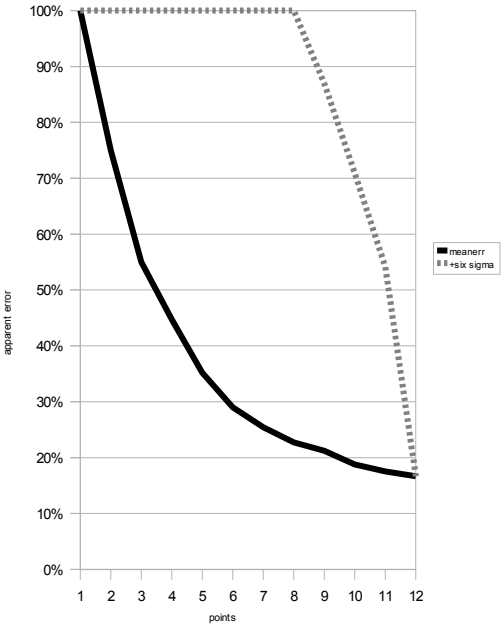

v-SVM - Linear Kernel




C-SVM - RBF Kernel

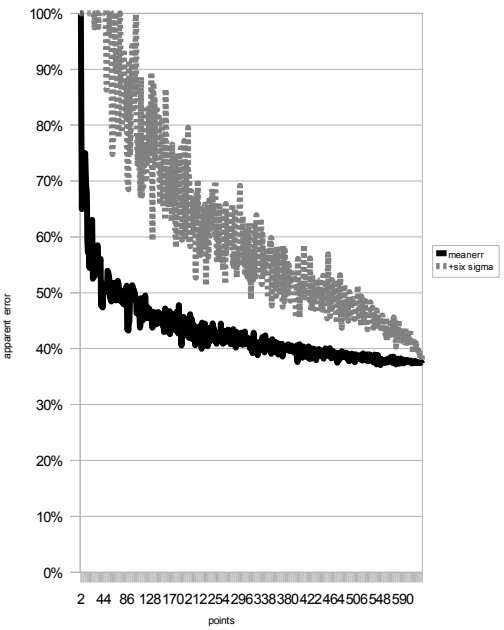



v-SVM - RBF Kernel

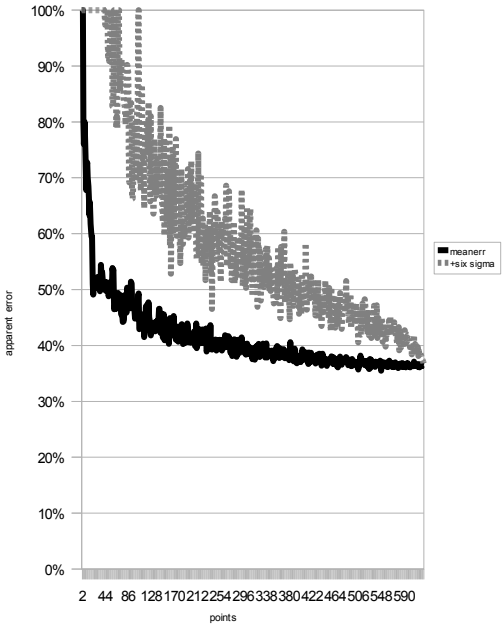


# K-NN

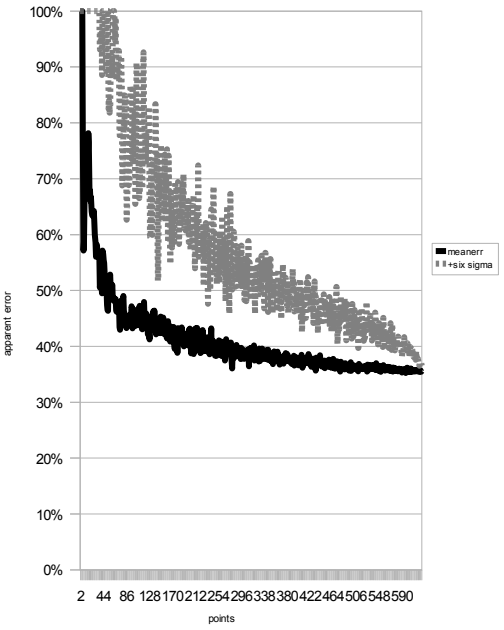
1-NN



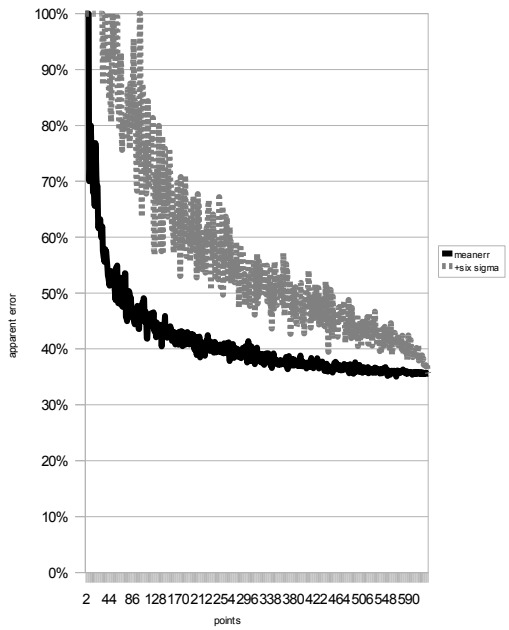

3-NN



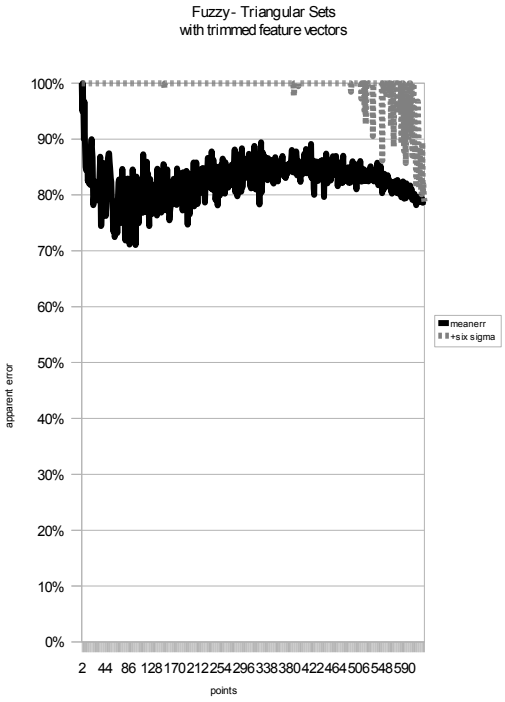

5-NN

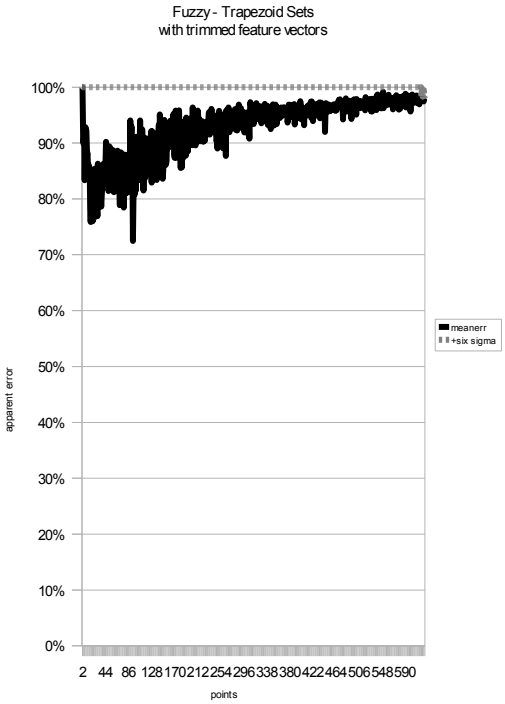


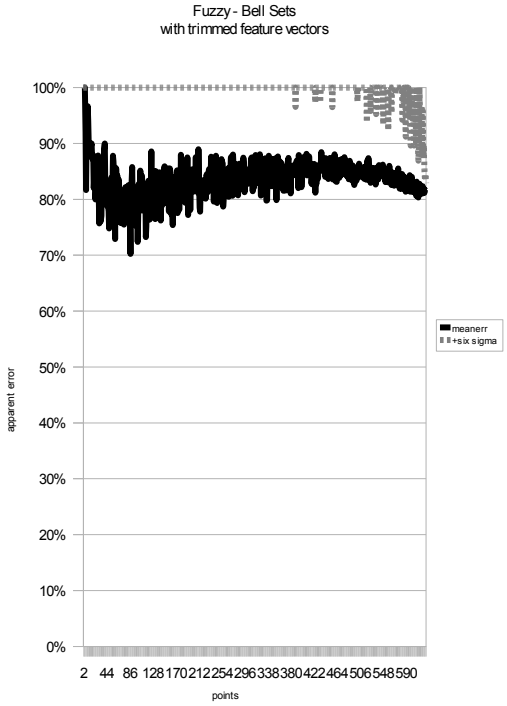

7-NN



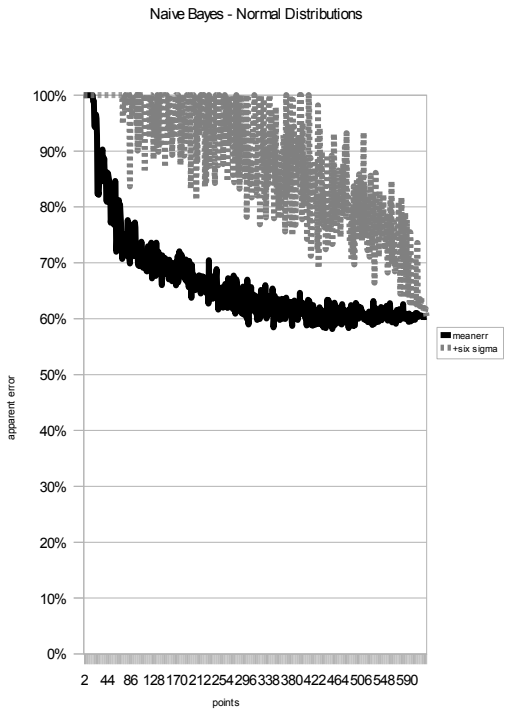
# Fuzzy

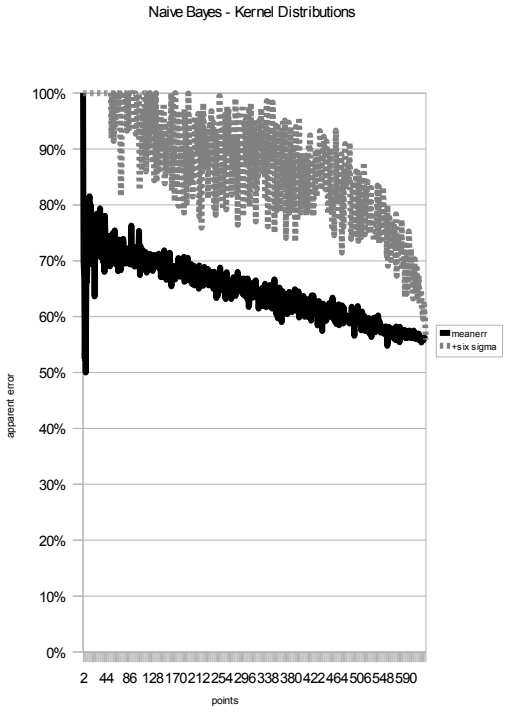

Fuzzy - Triangular Sets
with trimmed feature vectors
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
apparent error
2 44 86 128 170 212 254 296 338 380 422 464 506 548 590
points
meanerr
+six sigma

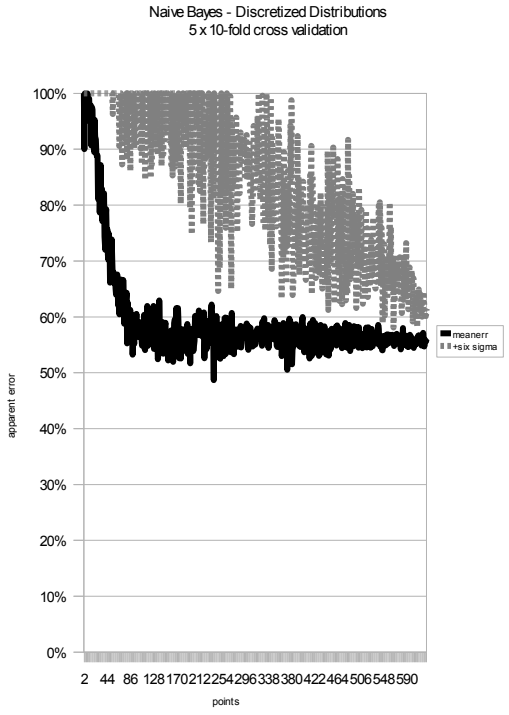

Fuzzy - Trapezoid Sets
with trimmed feature vectors
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
apparent error
2 44 86 128 170 212 254 296 338 380 422 464 506 548 590
points
meanerr
+six sigma


Fuzzy - Bell Sets
with trimmed feature vectors
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
apparent error
2 44 86 128 170 212 254 296 338 380 422 464 506 548 590
points
meanerr
+six sigma

# Naive Bayes

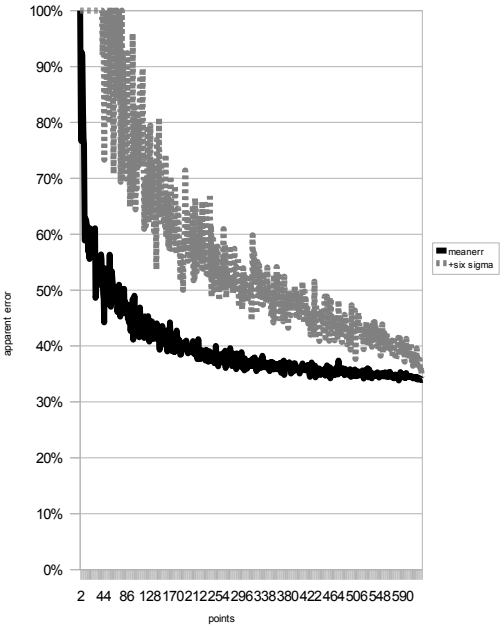

Naive Bayes - Normal Distributions
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
apparent error
2 44 86 128 170 212 254 296 338 380 422 464 506 548 590
points
meanerr
+six sigma


Naive Bayes - Kernel Distributions
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
apparent error
2 44 86 128 170 212 254 296 338 380 422 464 506 548 590
points
meanerr
+six sigma

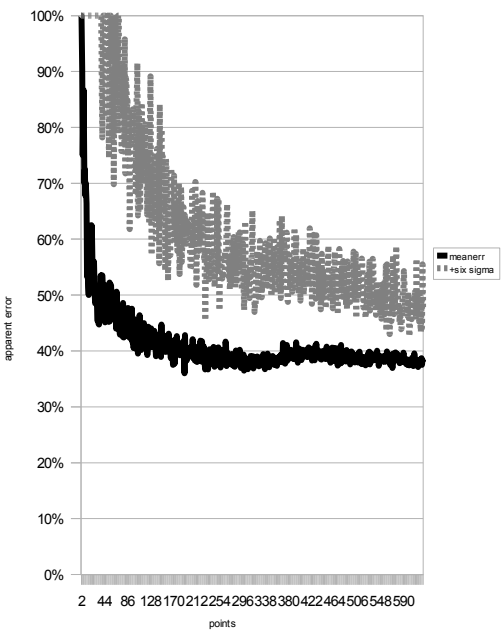

Naive Bayes - Discretized Distributions
5 x 10-fold cross validation
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
apparent error
2 44 86 128 170 212 254 296 338 380 422 464 506 548 590
points
meanerr
+six sigma

# SVM

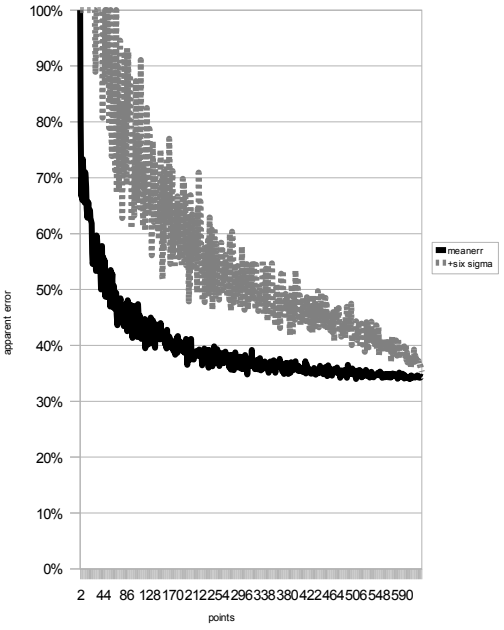
C-SVM - Linear Kernel



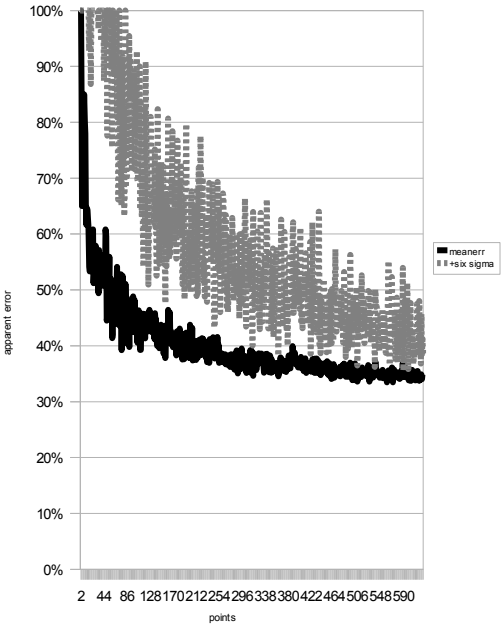

v-SVM - Linear Kernel




C-SVM - RBF Kernel

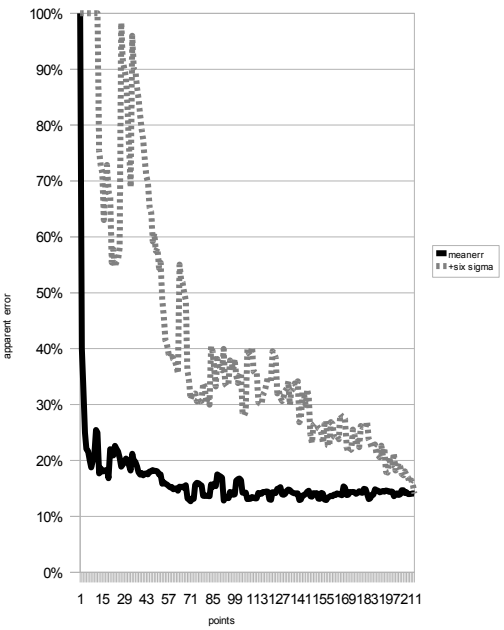



v-SVM - RBF Kernel

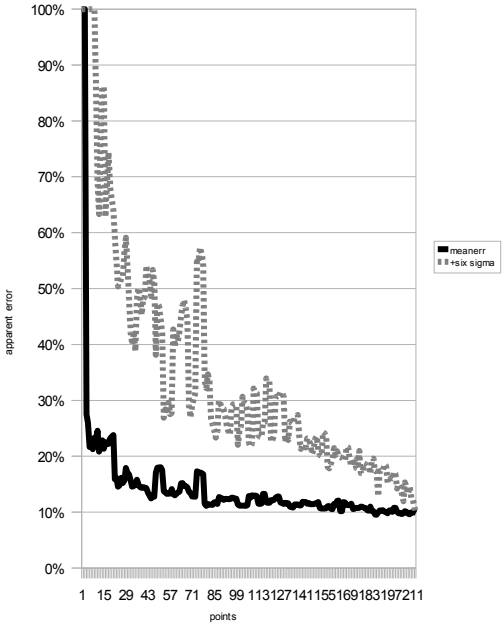


# K-NN

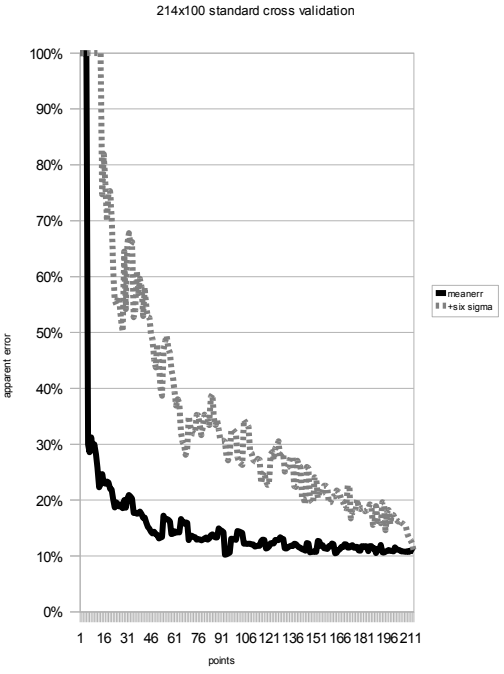
1-NN



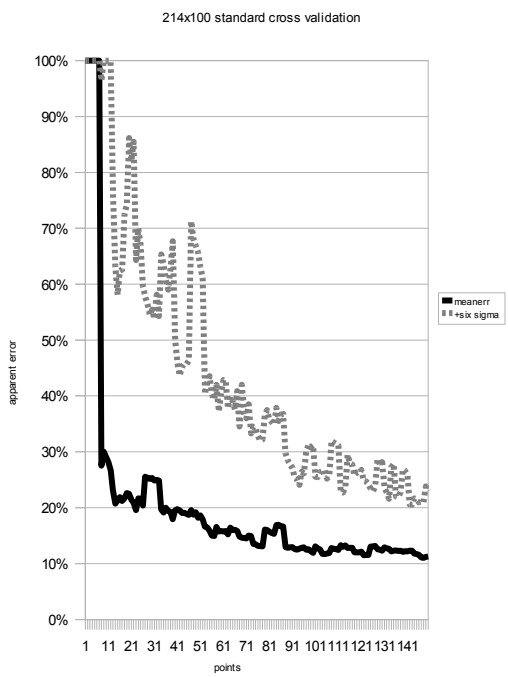

3-NN



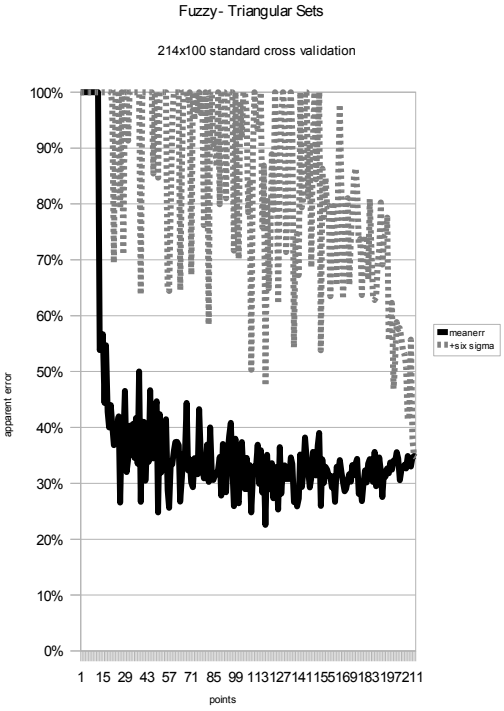

5-NN

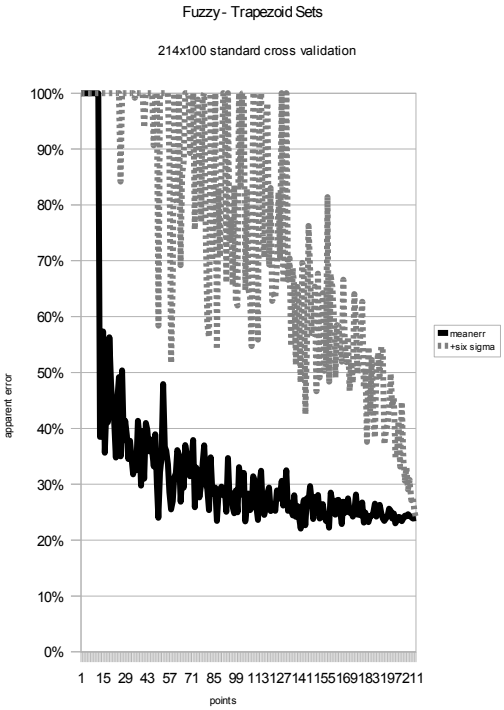


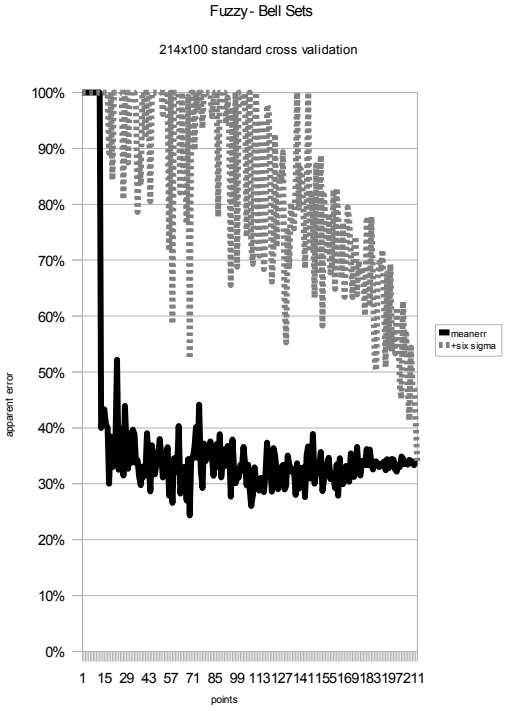

7-NN



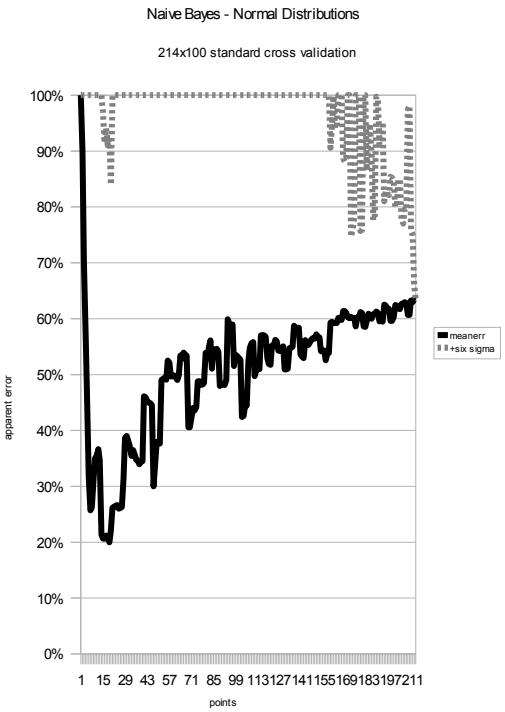
# Fuzzy

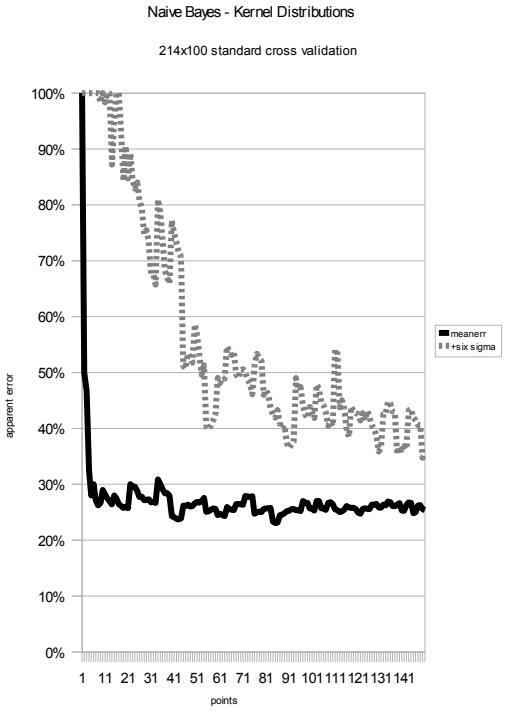

Fuzzy- Triangular Sets
214x100 standard cross validation
apparent error
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 15 29 43 57 71 85 99 113127141155169183197211
points
meanerr
+six sigma

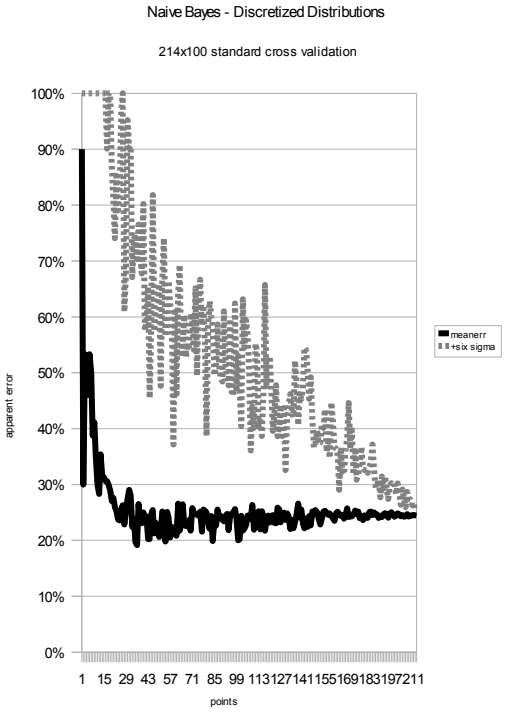

Fuzzy- Trapezoid Sets
214x100 standard cross validation
apparent error
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 15 29 43 57 71 85 99 113127141155169183197211
points
meanerr
+six sigma


Fuzzy- Bell Sets
214x100 standard cross validation
apparent error
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 15 29 43 57 71 85 99 113127141155169183197211
points
meanerr
+six sigma

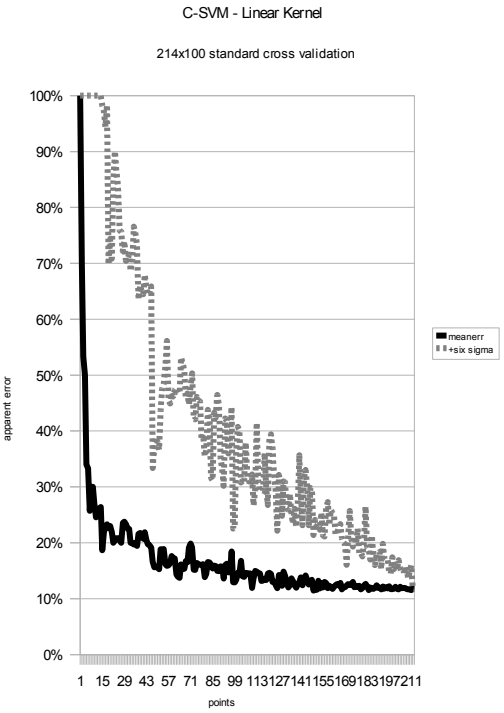
# Naive Bayes

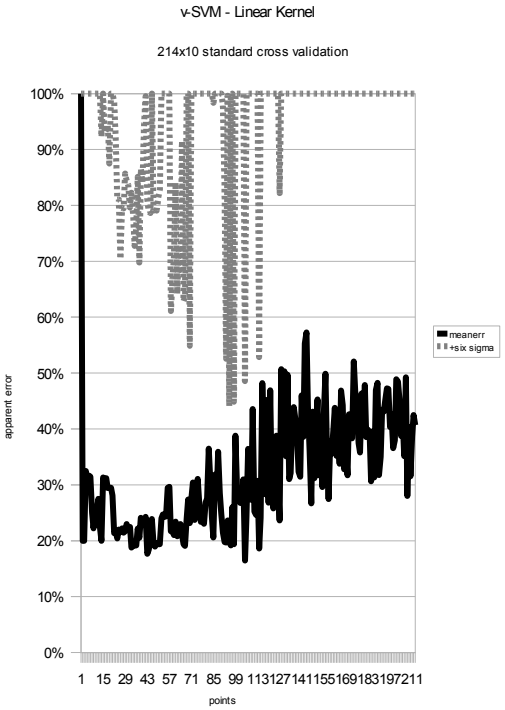

Naive Bayes - Normal Distributions
214x100 standard cross validation
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
apparent error
1 15 29 43 57 71 85 99 113127141155169183197211
points
meanerr
+six sigma

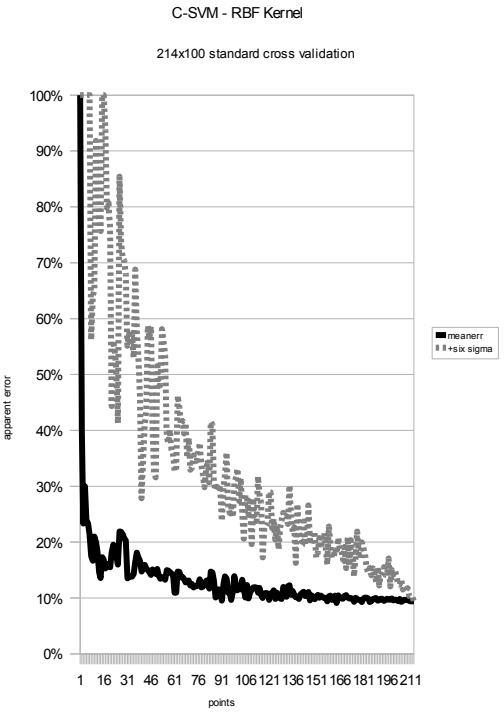

Naive Bayes - Kernel Distributions
214x100 standard cross validation
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
apparent error
1 11 21 31 41 51 61 71 81 91 101111121131141
points
meanerr
+six sigma

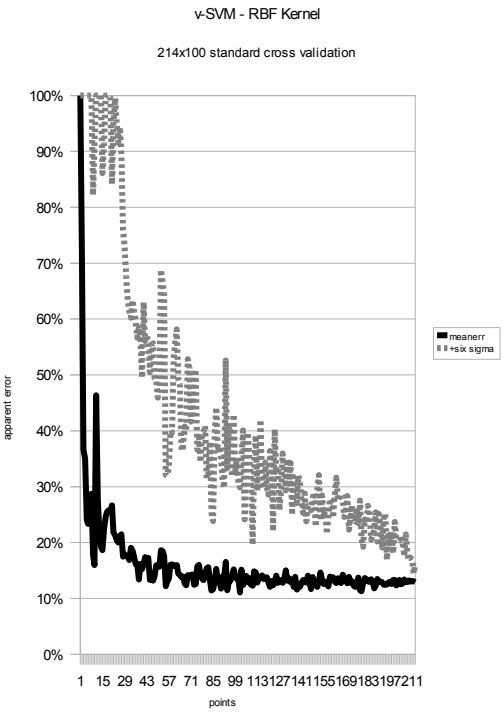

Naive Bayes - Discretized Distributions
214x100 standard cross validation
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
apparent error
1 15 29 43 57 71 85 99 113127141155169183197211
points
meanerr
+six sigma

# SVM


C-SVM - Linear Kernel
214x100 standard cross validation
apparent error
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 15 29 43 57 71 85 99 113127141155169183197211
points
meanerr
+six sigma


v-SVM - Linear Kernel
214x10 standard cross validation
apparent error
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 15 29 43 57 71 85 99 113127141155169183197211
points
meanerr
+six sigma


C-SVM - RBF Kernel
214x100 standard cross validation
apparent error
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 16 31 46 61 76 91 106121136151166181196211
points
meanerr
+six sigma


v-SVM - RBF Kernel
214x100 standard cross validation
apparent error
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 15 29 43 57 71 85 99 113127141155169183197211
points
meanerr
+six sigma

## A.4 CD-levyn sisältö


**bin** - Suoritettavat ohjelmatiedostot ja niiden tarvitsemat kirjastot.


**data** - Työssä käytetyt testiaineistot ja niiden käsittelyskriptit.


**casting** - ”Metallivalu” -testiaineisto csv -muodossa.


**glass** - ”Glass” -testiaineisto alkuperäisessä ja csv -muodossa.


**iris** - ”Iris” -testiaineisto alkuperäisessä ja csv -muodossa.


**imd** - ”Ruskuvalu” -testiaineisto raaka- ja csv -muodossa.


**paper** - ”Paperi” -testiaineisto csv -muodossa.


**docs** - Työn aikana tuotettu dokumentaatio, kuten tämä diplomityö ja referenssimanuaali.


**lib** - Työssä kaytettyjen valmiskirjastojen muokkaamattomat versiot


**src** - Työn aikana tuotettujen ohjelmistojen lähdekoodit ja niiden käyttämät valmiskirjastot.


**ANN** - valmiskirjasto binäärimuodossa.


**bayes** - Naiivi Bayes -luokitinkirjaston lähdekoodit.


**Classifier** - luokitinkehyksen lähdekoodit.


**ClassifierUI** - luokitinohjelmiston graafisen käyttöliittymän lähdekoodit.


**FeatureExtraction** - Halcon -konenäkökirjastoa käyttävä piirreirroitin IMD -aineistolle.


**FFLL** - Sumean päättelykoneen toteuttavan valmiskirjaston muokatut lähdekoodit.


**libsvm** - SVM -valmiskirjaston lähdekoodit.


**logger** - Luokittimien analysointityökaluja sisältävän apukirjaston lähdekoodit.


**PerformanceCUI** - luokitinohjelmiston komentorivikäyttöliittymän lähdekoodit.

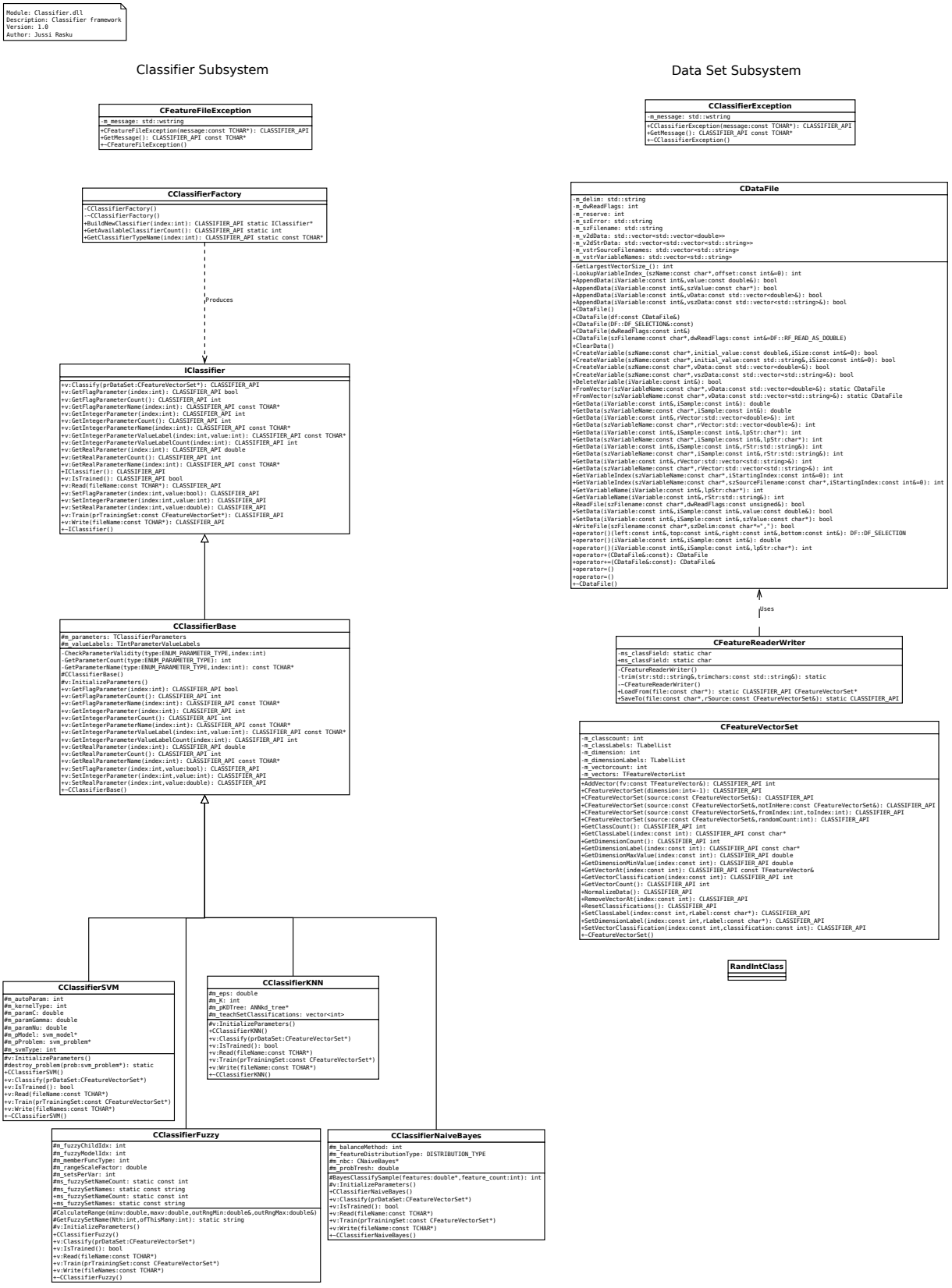

**tests** - Luokittimien testaamsiseksi suoritetut testit ja niiden tulokset.


**charts** - Testien tuloksista johdetut kuvaajat.


**results** - Testien tulokset ja niiden käsittely.


**scripts** - Testien ajamiseen käytetyt skriptit.

## A.5 Luokitteluohjelmiston UML-kaaviot



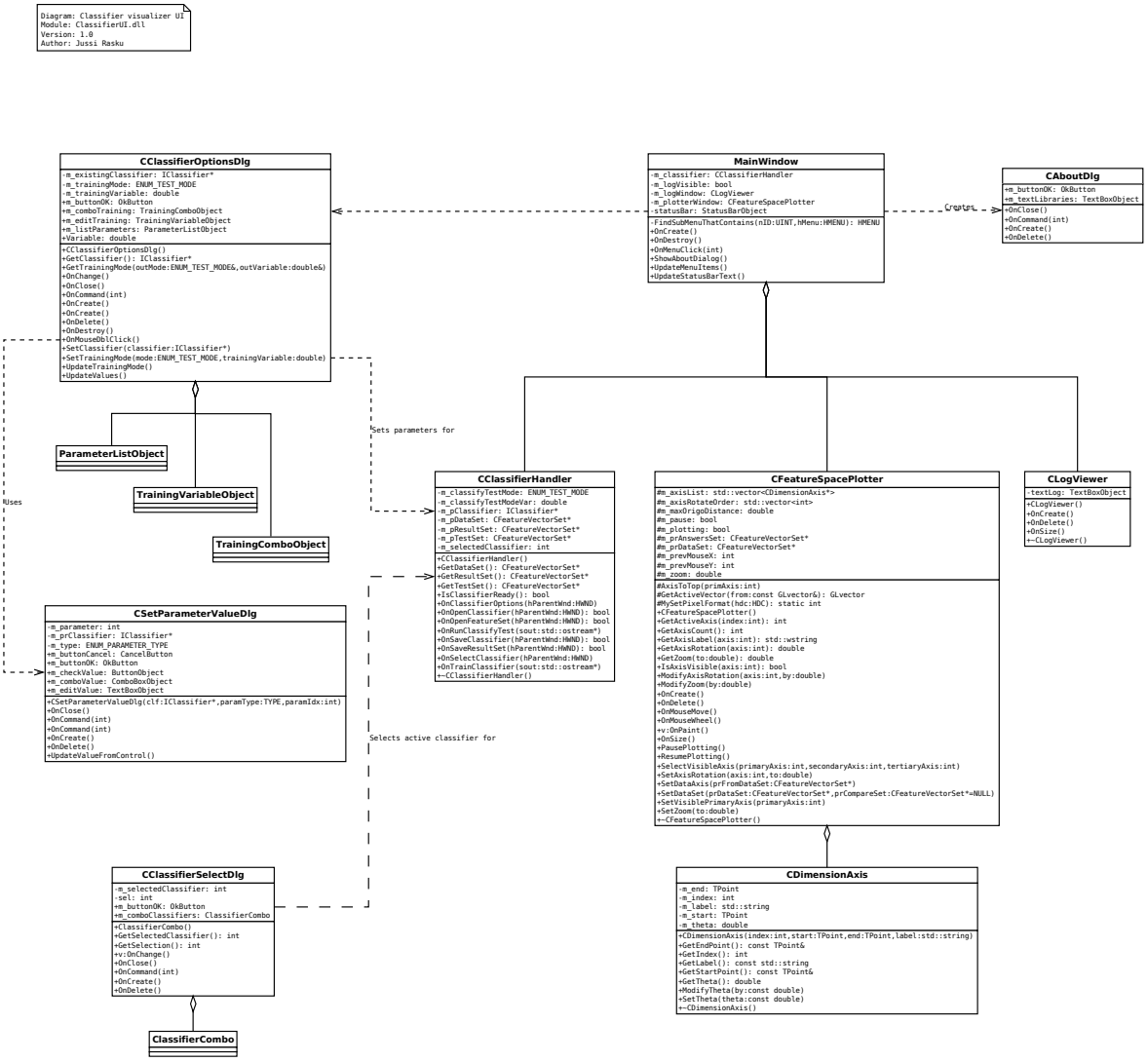


Diagram: Classifier visualizer UI
Module: ClassifierUI.dll
Version: 1.0
Author: Jussi Rasku
CClassifierOptionsDlg
-m_existingClassifier: IClassifier*
-m_trainingMode: ENUM_TEST_MODE
-m_trainingVariable: double
+m_buttonOK: OkButton
+m_comboTraining: TrainingComboObject
+m_editTraining: TrainingVariableObject
+m_listParameters: ParameterListObject
+Variable: double
+CClassifierOptionsDlg()
+GetClassifier(): IClassifier*
+GetTrainingMode(outMode:ENUM_TEST_MODE&,outVariable:double&)
+OnChange()
+OnClose()
+OnCommand(int)
+OnCreate()
+OnCreate()
+OnDelete()
+OnDestroy()
+OnMouseDblClick()
+SetClassifier(classifier:IClassifier*)
+SetTrainingMode(mode:ENUM_TEST_MODE,trainingVariable:double)
+UpdateTrainingMode()
+UpdateValues()
MainWindow
-m_classifier: CClassifierHandler
-m_logVisible: bool
-m_logWindow: CLogViewer
-m_plotterWindow: CFeatureSpacePlotter
-statusBar: StatusBarObject
-FindSubMenuThatContains(nID:UINT,hMenu:HMENU): HMENU
+OnCreate()
+OnDestroy()
+OnMenuClick(int)
+ShowAboutDialog()
+UpdateMenuItems()
+UpdateStatusBarText()
Creates
CAboutDlg
+m_buttonOK: OkButton
+m_textLibraries: TextBoxObject
+OnClose()
+OnCommand(int)
+OnCreate()
+OnDelete()
Sets parameters for
Uses
ParameterListObject
TrainingVariableObject
TrainingComboObject
CClassifierHandler
-m_classifyTestMode: ENUM_TEST_MODE
-m_classifyTestModeVar: double
-m_pClassifier: IClassifier*
-m_pDataSet: CFeatureVectorSet*
-m_pResultSet: CFeatureVectorSet*
-m_pTestSet: CFeatureVectorSet*
-m_selectedClassifier: int
+CClassifierHandler()
+GetDataSet(): CFeatureVectorSet*
+GetResultSet(): CFeatureVectorSet*
+GetTestSet(): CFeatureVectorSet*
+IsClassifierReady(): bool
+OnClassifierOptions(hParentWnd:HWND)
+OnOpenClassifier(hParentWnd:HWND): bool
+OnOpenFeatureSet(hParentWnd:HWND): bool
+OnRunClassifyTest(sout:std::ostream*)
+OnSaveClassifier(hParentWnd:HWND): bool
+OnSaveResultSet(hParentWnd:HWND): bool
+OnSelectClassifier(hParentWnd:HWND)
+OnTrainClassifier(sout:std::ostream*)
+~CClassifierHandler()
CFeatureSpacePlotter
#m_axisList: std::vector<CDimensionAxis*>
#m_axisRotateOrder: std::vector<int>
#m_maxDrigoDistance: double
#m_pause: bool
#m_plotting: bool
#m_prAnswersSet: CFeatureVectorSet*
#m_prDataSet: CFeatureVectorSet*
#m_prevMouseX: int
#m_prevMouseY: int
#m_zoom: double
#AxisToTop(primAxis:int)
#GetActiveVector(from:const GLvector&): GLvector
#MySetPixelFormat(hdc:HDC): static int
+CFeatureSpacePlotter()
+GetActiveAxis(index:int): int
+GetAxisCount(): int
+GetAxisLabel(axis:int): std::wstring
+GetAxisRotation(axis:int): double
+GetZoom(to:double): double
+IsAxisVisible(axis:int): bool
+ModifyAxisRotation(axis:int,by:double)
+ModifyZoom(by:double)
+OnCreate()
+OnDelete()
+OnMouseMove()
+OnMouseWheel()
+v:OnPaint()
+OnSize()
+PausePlotting()
+ResumePlotting()
+SelectVisibleAxis(primaryAxis:int,secondaryAxis:int,tertiaryAxis:int)
+SetAxisRotation(axis:int,to:double)
+SetDataAxis(prFromDataSet:CFeatureVectorSet*)
+SetDataSet(prDataSet:CFeatureVectorSet*,prCompareSet:CFeatureVectorSet*=NULL)
+SetVisiblePrimaryAxis(primaryAxis:int)
+SetZoom(to:double)
+~CFeatureSpacePlotter()
CLogViewer
-textLog: TextBoxObject
+CLogViewer()
+OnCreate()
+OnDelete()
+OnSize()
+~CLogViewer()
CSetParameterValueDlg
-m_parameter: int
-m_prClassifier: IClassifier*
-m_type: ENUM_PARAMETER_TYPE
+m_buttonCancel: CancelButton
+m_buttonOK: OkButton
+m_checkValue: ButtonObject
+m_comboValue: ComboBoxObject
+m_editValue: TextBoxObject
+CSetParameterValueDlg(clf:IClassifier*,paramType:TYPE,paramIdx:int)
+OnClose()
+OnCommand(int)
+OnCommand(int)
+OnCreate()
+OnDelete()
+UpdateValueFromControl()
Selects active classifier for
CClassifierSelectDlg
-m_seletedClassifier: int
-sel: int
+m_buttonOK: OkButton
+m_comboClassifiers: ClassifierCombo
+ClassifierCombo()
+GetSelectedClassifier(): int
+GetSelection(): int
+v:OnChange()
+OnClose()
+OnCommand(int)
+OnCreate()
+OnDelete()
ClassifierCombo
CDimensionAxis
-m_end: TPoint
-m_index: int
-m_label: std::string
-m_start: TPoint
-m_theta: double
+CDimensionAxis(index:int,start:TPoint,end:TPoint,label:std::string)
+GetEndPoint(): const TPoint&
+GetIndex(): int
+GetLabel(): const std::string
+GetStartPoint(): const TPoint&
+GetTheta(): double
+ModifyTheta(by:const double)
+SetTheta(theta:const double)
+~CDimensionAxis()

## A.6 Käsin rakennettu sumea sääntökanta

```
(* Casting dataset manual fuzzy sets and rules *)

FUNCTION_BLOCK

VAR_INPUT
        radius_ratio            REAL; (* RANGE(0.00 .. 1.00) *)
        perimeter_ratio         REAL; (* RANGE(0.00 .. 1.00) *)
        axis_ratio              REAL; (* RANGE(0.00 .. 1.00) *)
        approx_area             REAL; (* RANGE(0.00 .. 1.00) *)
END_VAR

VAR_OUTPUT
        class                   REAL; (* RANGE(0 .. 2) *)
END_VAR

FUZZIFY radius_ratio
        TERM small := (0.00, 0) (0.00, 1) (0.20, 1) (0.30, 0);
        TERM big := (0.20, 0) (0.30, 1) (1.00, 1) (1.00, 0);
END_FUZZIFY

FUZZIFY perimeter_ratio
        TERM small := (0.00, 0) (0.00, 1) (0.05, 1) (0.10, 0);
        TERM medium := (0.05, 0) (0.10, 1) (0.30, 1) (0.40, 0);
        TERM big := (0.30, 0) (0.40, 1) (1.00, 1) (1.00, 0);
END_FUZZIFY

FUZZIFY axis_ratio
        TERM small := (0.00, 0) (0.00, 1) (0.10, 1) (0.20, 0);
        TERM medium := (0.10, 0) (0.20, 1) (0.30, 1) (0.40, 0);
        TERM big := (0.30, 0) (0.40, 1) (1.00, 1) (1.00, 0);
END_FUZZIFY

FUZZIFY approx_area
        TERM small := (0.00, 0) (0.00, 1) (0.30, 1) (0.40, 0);
        TERM medium := (0.30, 0) (0.40, 1) (0.60, 1) (0.70, 0);
        TERM big := (0.60, 0) (0.70, 1) (1.00, 1) (1.00, 0);
END_FUZZIFY

FUZZIFY class
        TERM crack := 0;
        TERM hole := 1;
        TERM split := 2;
END_FUZZIFY

RULEBLOCK first
        AND:MIN;
        ACCU:MAX;
        RULE 1: IF (radius_ratio IS small) AND (perimeter_ratio IS small)
                 AND (axis_ratio IS small) AND (approx_area IS small)
                 THEN (class IS crack);
        RULE 2: IF (radius_ratio IS small) AND (perimeter_ratio IS small)
                 AND (axis_ratio IS medium) AND (approx_area IS small)
                 THEN (class IS crack);
        RULE 3: IF (radius_ratio IS small) AND (perimeter_ratio IS small)
                 AND (axis_ratio IS big) AND (approx_area IS small)
                 THEN (class IS crack);
        RULE 4: IF (radius_ratio IS big) AND (perimeter_ratio IS big)
                 AND (axis_ratio IS big) AND (approx_area IS big)
                 THEN (class IS hole);
        RULE 5: IF (radius_ratio IS small) AND (perimeter_ratio IS medium)
                 AND (axis_ratio IS small) AND (approx_area IS medium)
                 THEN (class IS split);
        RULE 6: IF (radius_ratio IS small) AND (perimeter_ratio IS medium)
                 AND (axis_ratio IS small) AND (approx_area IS big)
                 THEN (class IS split);
        RULE 7: IF (radius_ratio IS small) AND (perimeter_ratio IS medium)
                 AND (axis_ratio IS medium) AND (approx_area IS medium)
                 THEN (class IS split);
        RULE 8: IF (radius_ratio IS small) AND (perimeter_ratio IS medium)
                 AND (axis_ratio IS medium) AND (approx_area IS big)
                 THEN (class IS split);
END_RULEBLOCK

END_FUNCTION_BLOCK
```